AppDynamics App iQ Platform
AppDynamics Platform
Version 4.5.13
AppDynamics Essentials

Welcome to the AppDynamics Application Performance Monitoring (APM) Platform, version 4.5. AppDynamics helps you to understand and optimize the performance of your business, from its software to infrastructure to business journeys.

This documentation introduces you to the AppDynamics APM Platform. It describes concepts and procedures for the Platform that are common across the AppDynamics product modules, from Application Performance Monitoring to Application Analytics, End User Monitoring, and more. It also covers the Controller UI, the browser-based console you use to understand, tune, and troubleshoot your application environment.

To report issues with the documentation, email the AppDynamics docs team at docs@appdynamics.com.

New to AppDynamics?
- Getting Started
- AppDynamics Concepts
- Deploy AppDynamics
- Platform as a Service Integrations

Collaborating
- Manage Users and Groups
- Custom Dashboards
- Virtual War Rooms
- Health Rules

Product Homes
- Application Monitoring
- End User Monitoring
- Database Visibility
- Server Visibility
- Application Analytics

Other Resources
- Community Forums and Knowledge Base
- AppDynamics Help Center
- AppDynamics Exchange
- Support (login account required)
Getting Started

This topic gives you an overview of how to get started quickly with the AppDynamics Application Performance Management (APM) Platform, from getting a Controller to start deploying agents.

Get a Controller

The first step in using the AppDynamics App iQ Platform is to get a Controller. The Controller sits at the center of an AppDynamics deployment. It's where AppDynamics agents send data on the activity in the monitored environment. It's also where users go to view, understand, and analyze that data.

To get a Controller, start a free trial at www.appdynamics.com.

After creating your account and starting your trial, you can choose whether to install the Controller yourself or use a Controller hosted by AppDynamics.

- If you choose a SaaS Controller, AppDynamics hosts the Controller for you, and you do not need to install the Controller.
- If you choose an on-premises Controller instance, follow the instructions in the AppDynamics portal to download the Controller. You can download the Controller, like most installable components of the AppDynamics Application Performance Monitoring (APM) platform, from the AppDynamics download center at https://download.appdynamics.com/.

If you are investigating or learning about AppDynamics, you'll probably want to start with a SaaS Controller. It is the quickest way to get started. If you are ready for a closer look at AppDynamics, however, and interested in an on-premises deployment in the long term, see AppDynamics Application Performance Monitoring Platform for details on such implementations.

Log in to the Controller UI

Once you have installed the Controller or arranged for a hosted Controller, you can log in to the Controller UI.

- For a SaaS Controller, you will receive an email from the AppDynamics Operations team containing the URL and access details for your hosted Controller instance.
- For an on-premises Controller, log in using the administrator credentials you created during the installation.

Install AppDynamics Agents

Once you have a Controller, the next thing you do is to install AppDynamics agents in your environment.

The AppDynamics product module types use different types of agents. For example, for application monitoring, you need one of the AppDynamics App Agents.

For most types of agents, you can use the Agent Download Wizard in the Controller UI to get started quickly. The first time you log in to the Controller UI, the Agent Download Wizard appears. Follow the instructions in the wizard to install agents in your application environment. You can access the wizard directly from the home page in the Controller UI.

For more information on installing an app agent, use the following links:
Connect the Agent to the Controller

If you are using the agent download wizard, the wizard automatically configures the settings for connecting the agent to the Controller. If you download agents directly, you need to configure the Controller connections.

The following graphic shows the connections the agents use to reach the platform. Naturally, they vary depending on whether you are using the on-premises or SaaS AppDynamics platform.

It is possible that you will need to adjust the configuration of network components, such as firewalls or proxies, to permit traffic from the agent to the Controller. The connection is single directional, meaning that agents always initiate the connections to the Controller.

Enable Notifications (Configure an Email Server)

For the Controller to send notifications, such as those for health rule violations, the Controller needs to be configured to use a valid email (SMTP) server.

For a SaaS Controller, you can use the SaaS-provided SMTP Server or your SMTP Server. For an on-premises Controller, you need to configure an SMTP server after installation.

See Enable an Email Server for information on configuring an email server.

Roadmap for an AppDynamics Deployment
After you've installed agents and restarted your applications, AppDynamics automatically builds an environment of your application. You can see the model in the dashboards and flow maps view of the Controller UI.

As it monitors your application workload, AppDynamics determines what is normal for your environment and applies sensible defaults for detecting abnormal activity and application errors. You can likely start using and benefiting from AppDynamics immediately, without touching the instrumentation configuration. However, when you're ready, you can optimize the configuration to make the best use of AppDynamics for your environment and requirements.

The following timeline describes a typical getting started journey.

**First Day with AppDynamics**

1. Get a Controller
2. Download and install an AppDynamics agent.
3. Add users to the Controller UI.
4. Configure Email/SMS settings to enable notifications. For SaaS, AppDynamics preconfigures the SMTP server. For an on-premises Controller and SMS configuration, see Enable an Email Server.
5. Get to know the AppDynamics defaults, such as built-in health rules and baseline calculations.
6. Learn about the key performance indicator metrics.
7. Explore the ways that the different AppDynamics products work together.

**First Week with AppDynamics APM**

1. Create and organize your business applications.
2. View and customize flow maps.
3. Create custom dashboards.
4. Verify or adjust your tier groups, nodes, and remote services.
5. Customize your health rules.

**First Month with AppDynamics APM**

1. Tune your service entry points.
2. Manage user roles and groups, or connect the Controller to an external authentication provider.
3. Capture contextual information, information points, and data collectors.
4. Extend your health rules with custom actions.
5. Create custom metrics or add integration modules to AppDynamics.

**Tutorials**

Learn more about specific areas of the AppDynamics APM Platform by following one of these tutorials:

- Platform Installation Quick Start (on-premises installations only)
- Install the Java Agent
- Install the .NET Agent for Windows
- Manual Injection of the JavaScript Agent
- Instrument iOS Applications
- Instrument Android Applications
- Instrument Xamarin Applications
- Instrument Applications with the IoT C/C++ SDK
- Instrument Applications with the IoT Java SDK
- Instrument Applications with the IoT REST APIs
Download AppDynamics Software


Where applicable, AppDynamics also publishes software for distribution on package manager repositories such as RPM, npm, pip, NuGet, and others. For products that are available via the package manager, you can find relevant instructions to retrieve and install the software in the installation documentation.

When you download files, always copy or transfer the files in binary mode. If you want to transfer a file you have downloaded on Windows to a Linux machine, use binary mode in your transfer program when you move the file to the destination Linux environment.

Download AppDynamics Software with cURL

The following steps demonstrate how to use cURL to get the latest version of an AppDynamics software download:

1. Retrieve an OAUTH token with the download scope by running the following command:
   ```bash
   curl -X POST -d '{"username": "<username>","password": "<password>","scopes": ["download"]}’
   https://identity.msrv.saaas.appdynamics.com/v2.0/oauth/token
   ```
2. View the latest version of each available product from the URL https://download.appdynamics.com/download/downloadfilelatest/. For example,
   ```bash
   curl https://download.appdynamics.com/download/downloadfilelatest/
   ```
   Inspect the response (or parse via script) to find the `download_path` of the latest version of the product in which you are interested.
3. Download the binary by running the following command:
   ```bash
   curl -L -O -H "Authorization: Bearer <access_token>" <url_to_file>
   ```
   Replace `access_token` with your OAUTH token from step 1, and replace `url_to_file` with the `download_path` value from the response in step 3.

Validate Software Package Downloads

You can use checksum validation and, for specific packages, a digital signature to validate software you download from the AppDynamics download center.

These steps require the machine on which you validate a package to include a TLS 1.2 implementation.

Checksum Validation

Click the Checksums link beneath a software package on the AppDynamics download center to display the package's MD5 checksum and its SHA256 checksum. After your download completes, run a checksum tool and compare the results against the checksum information from the download center.

Digital Signatures

AppDynamics digitally signs the following packages using a certificate signed by a publicly known certificate authority:

- .NET Agent
- AppDynamics Controller for Windows MSI installer

AppDynamics digitally signs the following packages using a Pretty Good Privacy (PGP) key:

- Java Agent
- Standalone Machine Agent
- Standalone Machine Agent RPM package
- Python Agent pip package

The AppDynamics PGP public key is hosted on https://pgp.mit.edu under the User ID help@appdynamics.com. For information on using a PGP signature to validate a software package, see the tutorial on the AppDynamics Community Knowledge Base.
Lite and Pro Editions

On this page:

- AppDynamics Lite Features
- Transitioning from Pro Trial to AppDynamics Lite

You can try AppDynamics free for 15 days with AppDynamics Self-Service Pro Trial. At the end of the trial period, AppDynamics Self-Service Pro Trial transitions to the free product edition, AppDynamics Lite.

AppDynamics Lite Features

AppDynamics Lite provides the same application monitoring and troubleshooting features as AppDynamics Pro, with a few limitations as described here.

AppDynamics Lite includes the following license units:

- One unit of Java APM
- One unit of .NET APM
- One unit of PHP APM
- One unit of Node.js APM
- One unit of Python APM
- One unit of Web Server APM
- One unit of Database Monitoring
- Six units of Server Monitoring (via Standalone and .NET Machine Agents)

The data retention period is restricted to 24 hours.

For End User Monitoring, AppDynamics Lite provides access to a subset of features of the Pro product, as described in Browser RUM and Mobile RUM Licenses.

AppDynamics Lite restricts queries to the last 24 hours both from the user interface and via API. Also, you can manage five applications on AppDynamics Lite, as well as on AppDynamics Self-Service Pro Trial and Standard editions. To retain history for longer than 24 hours and monitor an unlimited number of applications, upgrade to AppDynamics Pro.

See http://www.appdynamics.com/pricing/us/ for information on changing or buying licenses.

Transitioning from Pro Trial to AppDynamics Lite

When the trial period ends, and AppDynamics detects an expired AppDynamics Pro Trial license, the Controller resets all agents. See information on resetting app agents in Manage App Agents.

After the reset, the Controller limits agent registration to one each of the Java Agent, the .NET Agent, the PHP Agent, and the Node.js Agent. The Controller permits the first agent of each type to register and use a license.

AppDynamics recommends you uninstall any agents you do not use. For instructions on how to uninstall an agent, see the uninstall topic corresponding to your agent type under Install App Server Agents.
Controller UI Overview

The AppDynamics Controller UI is the primary interface you will use to monitor, troubleshoot, and analyze your entire application landscape, from backend infrastructure and servers to the end user's app.

This topic introduces you to a few of the features of the Controller UI.

Supported Web Browsers

The AppDynamics Controller UI is an HTML 5-based browser application that works best with the latest version of any modern browser.

The Controller UI has been tested with and supports the last two versions of the following browsers:

- Safari
- Chrome
- Firefox
- Microsoft Edge
- Internet Explorer

A small number of pages in the Controller UI leverage Flash, and require Flash Player 10 or later; AppDynamics recommends version 11.

Certain types of ad blockers can interfere with features in the Controller UI. We recommend disabling ad blockers while using the Controller UI.

Navigating the UI

As you start using the Controller UI to configure AppDynamics and monitor your environment, it is helpful to get acquainted with the elements of the Controller UI that you will use most and that are referenced frequently in this documentation.

The precise controls and features that appear in the UI depending on the permissions of your user account and licensed product modules. For example, with Analytics licensed, you will have additional controls in application monitoring screens. See Manage Users and Groups for more information about user permissions.

The following image shows the main elements of the interface:
The UI elements include:

- **Top Navigation Bar**: Use to navigate the major product areas and cross product feature areas, such as Alert & Respond and Dashboard & Reports.
- **Left Navigation Menu**: Use to access sections specific to the product area.
- **Settings menu**: Access pages where you can customize the UI environment, view your account information, or, if permitted by your user role, configure administration settings for the UI such as add user accounts, configure authentication, and more.
- **Time range menu**: Controls what metric data AppDynamics represents in the UI. The UI only shows the data generated in the selected time range. See the following section for more information on the time range menu.

### Time Ranges in the UI

The Time Range menu appears in dashboards, the Metric Browser, and elsewhere in the UI. The time range selection determines what data appears in the main panel throughout the UI, except in the case of pages such as the Scalability Analysis page, where other time contexts apply. When you choose a new time range from the menu, it remains the active range as you navigate to other pages in the UI.

The time range menu has preconfigured time ranges you can choose from, or you can enter a custom time range. You can set custom time ranges that terminate in the past, making them useful for analyzing trends over time.

You can save custom time ranges as a named range for viewing later or for sharing with teammates. To save a time range, after you have selected a custom time range, click the Save icon next to the menu. Once saved, your custom named time range appears in the time range menu. Remove, modify, or share saved time ranges from the Manage Custom Time Range option accessible from the Time Range menu.

For a flow map, you can choose to enable the auto-refresh functionality only for 5 minutes and 15 minutes time range. From the standard time range list, select 5 or 15 minutes to view the Auto-refresh checkbox at the bottom of the list. Selecting the Auto-refresh checkbox refreshes data automatically for the chosen standard time range.

### Sharing Views With Other Users

Troubleshooting an application issue is often a collaborative effort involving multiple Controller UI users. You can share a particular view of the UI, such as a flow map or metric browser view, with other users using deep links.

A deep link is a shareable URL that retains the time range currently active in the UI. Time ranges that are relative to the current time, such as ‘last 15 minutes’, are represented as fixed time ranges in the URL.

To share a view, click **Settings > Copy a link to this screen to the clipboard**. You can then paste the URL where desired and share, for example in an email or chat window.
AppDynamics Mobile App

On this page:
- Supported Environments and Versions
- Get the Mobile App
- Configure your Account by Scanning a QR Code
- Configure your Account Manually
- Enable Notifications on the Device
- Enable Notifications from an On-Premises Controller

The AppDynamics mobile application lets you monitor the health and performance of your applications on Android and iOS devices. Alerts on a device can notify you of application issues within minutes of their occurrence.

Supported Environments and Versions

Using the AppDynamics Mobile App requires:
- Version 4.0 or later of the Controller
- For the device, these operating systems:
  - iPhone, iPad, or other iOS-based devices: iOS 8.0 or higher
  - Android: Android 4.0 (Ice Cream Sandwich) or higher

You can view custom dashboards on the mobile app, but node- or tier-level custom dashboards are not available for viewing in the Mobile App.

Get the Mobile App

To get the AppDynamics Mobile App:
- For an iPhone, search for AppDynamics in the Apple App Store or go there directly at AppDynamics Mobile App page.
- For an Android device, search for AppDynamics in Google Play.

After installation, you need to set up your account from within the application. There are two ways to do this:
- By scanning a QR code in the Controller UI with your mobile device.
- By configuring the Controller account settings manually.

The following section describes these ways to set up your account.

Configure your Account by Scanning a QR Code

To set up your account by QR code:

1. In the Controller UI, open your user preferences page. Click gear (⚙️) icon > My Preferences.
2. Click View Configuration Code next to Mobile App Account Configuration.
3. Follow the instructions shown to complete your device setup with the code.

Configure your Account Manually

1. After downloading and installing the app on your mobile device, launch the app.
   Click Add New Account if you have already set up an account and want to set up another.
2. Enter your Controller URL and click Next. If you are using SSL, make sure to check the Use HTTPS box.
3. Enter your account name based on your Controller type:
   - For a SaaS or multi-tenant Controller, enter the account name given to you when you signed up for a SaaS Controller or the account name added by your administrator for a multi-tenant Controller.
   - For a single-tenant on-premises Controller, use the built-in default account, Customer1.
4. Click **Next**.
5. Enter your username and password and click **Next**.

**Enable Notifications on the Device**

Push notifications enable you to stay on top of the health and performance of your applications without repetitively checking their status manually. Whether you are on the move or at the office, receiving notification of specific events happening within the system can mean minutes instead of hours of downtime. Our AppDynamics App lets you subscribe to any event that occurs on the Controller. To understand more about an AppDynamics event, see Monitor Events.

An on-premises Controller is not configured to generate push notifications by default. To enable push notifications, follow the instructions in Enable Notifications from an On-Premises Controller.

To subscribe to notifications on an iOS device:
1. In the AppDynamics mobile app, drill down to an application.
2. Open the application menu.

3. Select **Push Notification**.

4. Subscribe to events of interests for the selected app. Event types include health rule violation states, errors, application changes, and more:
To subscribe to notifications on an Android device:

1. In the AppDynamics mobile app, drill down to an application.
2. Open the overflow menu at the top right of the toolbar.
3. Select **Notifications Settings**.
3. Subscribe to events of interests for the selected app. Event types include health rule violation states, errors, application changes, and more.
If notification fails, AppDynamics captures the error in the Controller log.

To setup Push Notifications on an Android device:

1. Click **Settings** at the bottom of the main menu on the left side of the page.
2. Click **Push Notifications**. You will now receive notifications when health rule violations occur.
Enable Notifications from an On-Premises Controller

To generate notifications, the on-premises Controller must be able to reach the following address:

- [https://mobile-push.api.appdynamics.com/](https://mobile-push.api.appdynamics.com/)

You may need to configure your network or intervening firewalls to enable the Controller to access that domain.

Once you have ensured that the Controller can access the AppDynamics site, open the Administration Console. In the console find and set the `push.notification.service.enabled` flag to 'true'. The change takes effect immediately.
AppDynamics Support

Have a question about using AppDynamics or have you run into a problem? Try the following steps.

Search the Documentation

You can search for information from the following:

- The search field on the right side of the top menu bar initiates a search of the entire documentation set.
- The search field in the left navigation pane searches within the current version only.

You can search across AppDynamics information resources, including the community, knowledge base, support site, and more, from the AppDynamics Support Center.

Ask the Community

If you have questions about using AppDynamics, try asking the AppDynamics community.

Contact Support

If you need further assistance, contact your account representative or technical support.

For technical support, click the Help tab while logged in with your AppDynamics account in the AppDynamics Support Center.

When requesting support, attach relevant logs for your issue:

- For log files from the Controller, see Platform Log Files.
- For the heap, histogram, and thread dumps, see Controller Dump Files.
This page introduces you to the AppDynamics Application Performance Management (APM) Platform.

**About the AppDynamics APM Platform**

The AppDynamics APM Platform enables you to monitor and manage your entire application-delivery ecosystem, from the mobile app or browser client request through your network, backend databases and application servers and more.

AppDynamics APM gives you a single view across your application landscape, letting you quickly navigate from the global perspective of your distributed application right down to the call graphs or exception reports generated on individual hosts.

**Application Performance Monitoring**

At the tier level, AppDynamics gives you a view of the runtime operation of your code via an AppDynamics App Server agent. The agent detects calls to a service entry point at the tier and follows the execution path for the call through the call stack. It sends data about usage metrics, code exceptions, error conditions, and exit calls to backend systems to the Controller, either a SaaS or on-premises:

To get started with application monitoring, see Install App Server Agents.

Most application environments consist of more than a single application server. They may contain multiple, distributed, and interconnected servers and processes that participate in fulfilling a given user request. In this context, AppDynamics tracks transactions across distributed, heterogeneous services.
Infrastructure Visibility with Database Visibility

For greater visibility into your application delivery environment, you can add AppDynamics Database Visibility to the deployment.

App agents can tell you about calls to backend databases, including errors and call counts. The Database Visibility module extends that visibility to the workings of the database server itself. It gives you detailed information on query execution and performance, with an agent-less profile.

AppDynamics Server Visibility adds to your view of the data center, with precious information on the performance of the machines and networks in your environment.

In this deployment, the database agent collects information from the database servers and sends it to the Controller, which persists some of that information in the Events Service. Database analytics features may use the Events Service, the document storage component of the platform that AppDynamics has optimized for searching and storing high volumes of information.

End User Monitoring for Client Experience

While server-side monitoring can tell you a great deal about how end users experience your application’s performance and especially how to improve that performance from the server side, end-user monitoring can extend that insight all the way from the initial client request through to the response on the client device. With AppDynamics End User Monitoring, you can collect information on where in the world your requests are coming from and what devices and channels your users are using, as well as the performance of your code once it’s on your users’ devices. You can even investigate mobile crashes by seeing stack traces and other contextual data at the moment of the crash, and AppDynamics can tie that data to business transaction data from the server side.

Business iQ and Application Analytics for Business Impact
How does the overall performance of your application environment affect your business? Business iQ, powered by AppDynamics Application Analytics, can help you understand how the performance of your application environment and end-user applications ties to the business data of the transactions. It lets you sort, order, and understand the data that compose the business transactions. It also enables you to drill into the varieties of log data that your environment generates. See Using Application Analytics Data for information about how to install and use application analytics.

Using Metrics

A metric is a particular class of measurement, state, or event in the monitored environment. Many defaults relate to the overall performance of the application or business transaction, such as request load, average response time, or error rate. Others describe the state of the server infrastructure, such as percentage CPU busy or percentage of memory used.

Agents register the metrics they detect with the Controller. They then report measurements or occurrences of the metrics (depending on the nature of the metric) to the Controller at regular intervals. You can view metrics using the Metric Browser in the Controller UI.

Information points are a particular type of metric that enables you to report on how your business (as opposed to your application) is performing. For example, you could set up an information point to total the revenue from the purchase on your website of a specific product or set of products. You can also use information points to report on how your code is performing, for example, how many times a specific method is called and how long it is taking to execute.

You can create extensions that use the machine agent to report custom metrics that you define. These metrics are base-lined and reported in the Controller, just like the built-in AppDynamics metrics.

As an alternative to using the Controller UI, you can access metrics programmatically with the AppDynamics APIs.

Baselines and Thresholds

The AppDynamics Platform uses both self-learned baselines and configurable thresholds to help identify application issues. A complex distributed application has a large number of performance metrics, and each metric is important in one or more contexts. In such environments, it is difficult to:

- Determine the values or ranges that are normal for a particular metric
- Set meaningful thresholds on which to base and receive relevant alerts
- Determine what is a 'normal' metric when the application or infrastructure undergoes change

For these reasons, anomaly detection based on dynamic baselines or thresholds is one of the essential features of the AppDynamics platform.

The AppDynamics platform automatically calculates dynamic baselines for your metrics, defining what is 'normal' for each metric based on actual usage. Then the platform uses these baselines to identify subsequent metrics whose values fall out of this normal range. Static thresholds that are tedious to set up and, in rapidly changing application environments, error-prone, are no longer needed.

You can create health rules with conditions that use baselines, allowing you to trigger alerts or kick off other types of remedial actions when performance problems are occurring or may be about to happen. See Alert and Respond and Health Rules and Dynamic Baselines for more detail.

AppDynamics thresholds help you to maintain service level agreements (SLAs) and ensure optimum performance levels for your system by detecting slow, very slow and stalled transactions. Thresholds provide a flexible way to associate the right business context with a slow request to isolate the root cause. See Transaction Thresholds.

Health Rules, Policies, and Actions

AppDynamics uses dynamic baselining to establish what is considered normal behavior for your application automatically. Then you can set up health rules against those standard baselines (or use other health indicators) to track non-optimal conditions. A health rule might be 'create a critical event when the average response time is four times slower than the baseline.'

Policies allow you to connect such problematic events (like the health rule critical event) with actions, that can, for example, trigger alerts or remedial behavior, addressing the system's issues before your users are affected.

AppDynamics supplies default health rules. You can customize the default health rules and create new rules specific to your environment.

The out-of-the-box health rules test business transaction performance as follows:
- **Business Transaction response time is much higher than normal**: Defines a critical condition as the combination of an average response time higher than the default baseline by three standard deviations and a load greater than 50 calls per minute. This rule defines a warning condition as the combination of an average response time higher than the default baseline by two standard deviations and a load greater than 100 calls per minute.

- **Business Transaction error rate is much higher than normal**: Defines a critical condition as the combination of an error rate greater than the default baseline by three standard deviations and an error rate higher than ten errors per minute and a load greater than 50 calls per minute. This rule defines a warning condition as the combination of an error rate greater than the default baseline by two standard deviations and an error rate greater than five errors per minute and a load greater than 50 calls per minute.

For more information, see [Alert and Respond](#).

## Infrastructure Monitoring

While business transaction performance is the typical focus of a performance monitoring strategy, monitoring infrastructure performance can add insight to underlying factors in business transaction performance. AppDynamics can alert you to the problem at the business transaction level and the infrastructure level.

AppDynamics provides preconfigured application infrastructure metrics and default health rules to enable you to discover and correct infrastructure problems. You can also configure additional persistent metrics to implement a monitoring strategy specific to your business needs and application architecture.

In addition to health rules, you can view infrastructure metrics in the Metric Browser. In this context, the Correlation Analysis and Scalability Analysis graphs can be particularly useful for understanding how infrastructure metrics can correlate or relate to business transaction performance.

## Integrating and Extending AppDynamics

AppDynamics provides many ways for you to extend AppDynamics Pro and integrate metrics with other systems. The [AppDynamics Exchange](#) contains numerous extensions you can download, and you can develop your own if you can't find what you need.

Extensions for AppDynamics come in the following categories:

- **Monitoring Extensions** add metrics to the existing set of metrics that AppDynamics agents collect and report to the Controller. These can include metrics that you obtain from other monitoring systems. They can also include metrics that your system extracts from services that are not instrumented by AppDynamics, such as databases, LDAP servers, web servers, or C programs. To write specific monitoring extensions, see [Extensions and Custom Metrics](#).
- **Alerting Extensions** let you integrate AppDynamics with external alerting or ticketing system and create custom notification actions. To learn how to write specialized custom notification see [Build a Custom Action](#). Also, see [Email Templates](#) and [HTTP Request Actions and Templates](#).
- **Performance testing extensions** consist of performance-testing extensions, such as the one described on [Integrate AppDynamics with Apica](#).
- **Built-in integration extensions** are bundled into the AppDynamics platform and only need to be enabled or configured. These include:
  - Integrate AppDynamics with Splunk
  - Integrate AppDynamics with DB CAM

For creating custom extensions and integration components for AppDynamics, see the AppDynamics [API information](#).
Deploy AppDynamics

On this page:
- About AppDynamics Deployment
- Deployment Models
- Network Port Requirements
- Deployment Planning Roadmap

The topics in this section describe considerations for deploying AppDynamics, from available architectural models for the platform to an overview of AppDynamics agent installation and administration.

About AppDynamics Deployment

When starting out with AppDynamics, you may want to try deploying it in a small-scale demonstration environment. Deploying AppDynamics to a full production environment introduces complexity, such as increased scale, with possibly hundreds of applications to monitor, non-traditional running environments such as Platform-as-a-Service (PaaS) or virtualized systems, and complex network topologies.

AppDynamics was designed to handle this complexity, with support virtualized and PaaS systems and tools for deploying and maintaining large-scale AppDynamics agent deployments.

Deployment Models

You can deploy the AppDynamics Platform in the following forms:

- In a SaaS deployment, AppDynamics manages the server-side components of the AppDynamics Platform, including their installation and upgrades. You only need to install and manage agent-side components.
- In an on-premises deployment, you install and manage all platform components.

The deployment type that is best for you depends on your requirements. For example, if you require all data to remain within your own IT infrastructure, an on-premises deployment is appropriate. If your data can be located in the cloud, you might consider using a SaaS deployment instead.

For information on on-premises installations, see AppDynamics Application Performance Monitoring Platform.

Network Port Requirements

To deploy AppDynamics, you may need to modify the configuration of network components to permit access to ports used in the AppDynamics deployment. The following pages describe the specific ports used in the system:

- For SaaS: SaaS Domains and IP Ranges
- For the on-premises platform: Port Settings

Deployment Planning Roadmap

See Getting Started to get started with a trial or investigatory installation of AppDynamics. For a minimal installation, you can easily get AppDynamics up and running in an hour or two. Deploying to a production environment, however, usually adds considerations to the initial installation process. You might have security, network, and change management requirements governed by your organizational policies.

This section outlines the typical production roadmap and gives you links to where you can get more information about the individual steps. It is divided into the following broad categories. Depending on your goals, the size of the deployment, and other factors, you may wish to combine steps or break specific steps out further, as best makes sense for your environment.
1. Plan the Controller and Platform Installation (on-premises)
2. Plan for Security
3. Define Controller operating procedures (on-premises)
4. Devise Agent rollout strategy
5. Plan AppDynamics Model
6. Users who will access AppDynamics
7. Define monitoring strategy

Plan the Controller and Platform Installation (on-premises)

1. Determine your deployment profile, small, medium, or large.
2. Follow the hardware sizing guidelines for your profile.
3. Discover network layout, bandwidth, and connectivity. View the port connections required for the AppDynamics components you are installing, and address network requirements to permit the connections.

Plan for Security

1. Determine the access control strategy for UI users, including locally authenticated users and externally authenticated users (LDAP, SAML, and so on).
2. Determine SSL requirements and key management.

Define Controller operating procedures (on-premises)

1. HA setup (via Enterprise Console)
2. Failover methodology (scripted, automated)
3. Data backup strategy (hot, cold or LVM, frequency, the storage required, metadata backup)

Devise Agent rollout strategy

1. Manual, automated or scripted, or Universal Agent
2. Integrating rollout with your application deployment process

Plan AppDynamics Model

1. Understand how your application maps to the AppDynamics concept of application, tier, and node. Determine naming strategy for the model.
2. Devise Business Transaction strategy. Determine which end-to-end business processes in your environment you want to monitor as Business Transactions.
3. Devise End User Monitoring and Analytics data collection strategy.
4. Consider database and hardware monitoring strategy.

Users who will access AppDynamics

1. Identify monitoring team or others who will administer AppDynamics. Ensure that they can maintain it, both from a resource and skill-set standpoint.
2. Identify stakeholder from each group that will use AppDynamics.

Define monitoring strategy

1. Assess the alert and health rule strategy that makes sense for your requirements.
2. Define reporting strategy.
3. Determine integration points with other systems (using extensions or REST API).
Platform as a Service Integrations

The AppDynamics Application Performance Monitoring (APM) Platform integrates with Platform as a Service (PaaS) providers, providing easily implemented AppDynamics monitoring for cloud or PaaS-hosted applications.

In general, the integration allows developers to add AppDynamics app agent instrumentation to their applications as additional services when provisioning their application host instance. In some cases, AppDynamics can provide machine-level monitoring information for the virtual platform and supporting infrastructure.

The following section lists the PaaS providers with which the AppDynamics platform integrates. For details on how to use the AppDynamics specific platforms, see the documentation listed.

Platform as a Service (PaaS) Provider Support

<table>
<thead>
<tr>
<th>PaaS Provider</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pivotal Cloud Foundry (PCF)</td>
<td>Application and machine monitoring in PCF environments. AppDynamics can provide built-in support for these Cloud Foundry buildpacks:</td>
</tr>
<tr>
<td></td>
<td>• <strong>Java Buildpack 3.4 and higher.</strong> For a walkthrough of using the Java buildpack, see Using AppDynamics with Java Applications on Pivotal Cloud.</td>
</tr>
<tr>
<td></td>
<td>• <strong>PHP Buildpack 4.0+.</strong> For a walkthrough of using the PHP buildpack, see Using AppDynamics with PHP Applications on Pivotal Cloud Foundry.</td>
</tr>
<tr>
<td></td>
<td>For more information and links to the AppDynamics tiles, see the AppDynamics PCF documentation.</td>
</tr>
</tbody>
</table>

| Red Hat OpenShift 3           | Docker images with built-in AppDynamics monitoring support for JBoss EAP 6.4 and WildFly 8.1. For documentation and download information, see the AppDynamics Java APM Agent page on the Red Hat Customer Portal. |

| Microsoft Azure               | You can use NuGet or the Windows Azure Portal to install the AppDynamics .NET Agent package directly to projects running on Windows Azure. You can instrument Web Roles, Worker Roles, and App Services. See Deploy AppDynamics for Azure (Moving). |
License Management

On this page:

- Permissions
- License Management Scope
- Viewing License Information
- Viewing Account Usage
- Creating Rules
- Viewing Access Keys

Related Pages:

- License Entitlements and Restrictions
- Browser RUM Licenses
- Mobile RUM Licenses

The AppDynamics license that you purchase determines the features and deployment scale available to you. For all AppDynamics Application Performance Monitoring (APM) Platform deployments except the on-prem EUM Server and on-prem Synthetic Server, the license resides on the Controller. For the on-prem EUM Server and Synthetic Server, the EUM license resides on the on-prem EUM Server.

There are two different types of AppDynamics Licenses: Pro and Lite. You can try the Pro license free for 15 days. When the free trial ends, the license converts to a Lite license, as described in Lite and Pro Editions.

When you purchase Application Performance Monitoring licenses, you can create rules to fine-tune license distributions across your organization.

Permissions

Users must have a role with the View and Configure Licenses permission for this activity.

License Management Scope

The AppDynamics Controller uses licensing to determine the features that are licensed and available in an AppDynamics Application Performance Monitoring (APM) Platform deployment and the scale of the deployment. The License Management module provides visibility into your license consumption model, allows you to manage entitlements from your license pools, and manage and distribute your access keys usage so you can scale out your different applications.

The following illustrates the flexibility of license assignments under the license scope configuration model.
License Management Scope configuration allows you to assign licenses to specific applications and machines. Using scopes to configure licenses allows you to express and scale the configuration modules as required.
License Management provides:

- Granular control with precise rules and access keys to restrict which hosts and apps consume licenses
- Great visibility with detailed current and historical usage reporting across business units and services

On the Add and Edit Rule dialogs, you can create or edit rules to specify license allocation according to selected agents, applications, and machines/servers, following a mix and match model. For example, a rule can be applied to a selected application running on several specific machines.

Viewing License Information

To view license information in the Controller UI, click the gear icon ⚙️ and choose License.

The License dashboard consists of three tabs:

- **Account Usage**: displays the maximum license usage in the time range you specify
- **Rules**: allows you to configure license allocation across machines and applications
- **Account**: displays the account name, access key, and expiration

The following sections cover each of the tabs in more detail.

Viewing Account Usage

In the Account Usage tab, you can view the number of licenses consumed and license expiration dates for the following product areas:

- Applications: Java, .NET, C/C++, Apache, PHP, Node.js, Python, Go
- User Experience
You can click on each application type to see that type's license usage. You can specify the time range that you want to see data for in the drop-down menu on the upper right. For more information about how AppDynamics defines license units for each product module, see License Entitlements and Restrictions.

Creating Rules

In the Rules tab, you can create rules to specify the number of Application Performance Monitoring licenses to allocate to specified applications and machines. If you choose not to create any rules, your license allocation follows the default rule, which distributes the licenses across all applications and machines.

In a rule, you can specify the following allocation details:

- How many units you want to allocate to each rule
- Which applications are allowed to consume the units (agents) allocated by this rule
- Which servers are allowed to consume the units (agents) allocated by this rule

When license rules are enabled, you must ensure that all your units are accounted for by either the default rule or a custom rule. AppDynamics follows a mix and match model: you can adjust the number of units allocated to each license rule at any time.

Create a Custom Rule

1. Click the gear icon and choose License.
2. Enable the license rules feature by clicking Create Rules on the Rules tab.
3. Click +Create to display the Add Rule dialog box.
4. In the General tab, enter the name of your rule and the number of units for each agent module type that this rule allocates.
5. In the Application Scope tab, select Specified Applications to restrict which applications agents can report to.
6. Only agents that report to the applications you specify can consume the units allocated by this rule. You can select one or more applications from the Available Applications list and click the left arrow button to move the applications to the Selected Applications box. You can specify matching criteria for allowed applications by clicking Add. You can also specify matching criteria for applications that you have not yet created. The applications you specify using matching criteria will not appear in the Selected Applications box.
7. You can select up to 100 applications to assign to a rule.
8. In the Server Scope tab, select Specified Servers to restrict which servers agents can be deployed to.
9. Only agents deployed on the servers you specify can consume the units allocated by this rule. You can select one or more servers from the Available Servers list and click the left arrow to move the applications to the Selected Applications box. You can specify matching criteria for allowed servers by clicking Add. You can also specify matching criteria for servers not yet monitored by AppDynamics. The applications you specify using matching criteria will not appear in the Selected Applications box.
10. Using the Also include applications/servers matching the following criteria option, you can select applications and servers by matching multiple machines using the match criteria. This is more efficient than scrolling through thousands of applications and machines to make your selections manually.
11. You can also add uninstrumented applications to the rule in the Application Scope and unmonitored machines in the Server Scope, so that when you install the agents to instrument these unmonitored applications and servers, they will be allowed to use the licenses defined in this rule.

After You Create a Rule

After creating your custom rule, return to the default rule and decrement your unit allocation by the number of units that you allocated in your custom rule. Otherwise, the allocations will show as over-provisioned.

For each rule you create, a new access key is generated, and the agents under that rule are authenticated with that generated key. Update your agent to use this new access key and restart the agent.

You must allocate the same number of machine agents as your APM agents. For example, if you allocate ten Java agents,
Viewing License Usage

To view license usage for the units allocated by a rule, click the rule. AppDynamics indicates license usage percentage by color:

- **Blue:** under 80% units used
- **Yellow:** over 80% units used
- **Red:** 100% units used

Over-provisioning Units

When configuring rules, you can allocate more units than you were originally provisioned by AppDynamics. Over-provisioning units buffer your unit allocation and is useful if you are unsure which agent type will consume more units. While your unit allocation is unbounded, AppDynamics limits your unit usage according to the number of licenses you purchase. In other words, AppDynamics continues to report data for over-provisioned units as long as you have the units available, such as when the license usage for certain agents is lower than expected.

If a Licensed Agent cannot Register with the Controller

If a licensed agent such as a Server Visibility cannot register with the Controller, do the following:

- Check that the user account has a product license for the relevant agent type.
- If the user account has license rules defined, make sure these have the correct number of license units allocated. To change
the number of allocated units in a rule:

1. Go to Controller Settings (gear icon ➕) > License > Rules.
2. Edit the License Rule of interest. (There might be only one License Rule, named Default.)
3. In the General tab, set the Allocated Units field for the relevant license and apply the change.

License Rules Best Practices

When creating license rules, you may want to follow consistent criteria for selecting which applications belong to a rule. We recommend that you follow one of these criteria:

- Group applications by business unit. If you want a rule to contain multiple applications, select applications that belong to the same business unit.
- If you want to view individual application usage, select only one application for a rule.

Viewing Access Keys

On the Account tab, you can view your license type (Pro or Lite) and access key. You use the access key to connect agents to the Controller. See Agent-to-Controller Connections.
Applying or Updating a License File

On this page:
- Applying a License
- Applying a License to a Multi-Tenant Controller
- Updating a License
- Updating License Rules
- Updating MAC Address

Related pages:
- Application Analytics Licenses

For AppDynamics Pro trial installations and SaaS Controllers, the license is applied and installed automatically.

If you are using AppDynamics on-premises, you must manually apply the license file after installing the Controller. Whenever you purchase additional units, you must update the license file. If you are updating Application Analytics licenses in an on-premises environment, you may have to do some of the updates manually, once you have updated your Controller license. See Application Analytics Licenses for more information.

Applying a License

1. Copy the license file to the Controller installation directory. After copying the license file, allow up to five minutes for the license change to take effect.
2. To verify that your license file is applied, check that the license you purchased appear in the Peak Usage tab of the License page.

Applying a License to a Multi-Tenant Controller

If you are updating the license file for a multi-tenant Controller, you must manually edit the accounts to affect the change. For example, if you purchase an additional 200 licenses to distribute across two accounts, you must manually allocate those other units by updating each account.

1. Copy the license file to the Controller installation directory. After copying the license file, allow up to 5 minutes for the license change to take effect.
2. Login with root user at http://<controller_host>:<port>/controller/admin.jsp
3. Click Accounts and select the account you want to apply the license to.
4. Update the license expiration date and license units based on license.lic.
5. Save the changes.

Updating a License

1. Rename your old license file to something other than license.lic, so that the Controller ignores it.
2. Copy the new license file to the Controller installation directory. After copying the license file, allow up to 5 minutes for the license change to take effect.
3. To verify that your license file is applied, check that the license you purchased appears in the Account Usage tab of the License page.

Updating License Rules

If you are using license rules, you must update them to account for the additional units you purchased, after you update your license. For example, if you buy another 200 licenses to distribute across two license rules, you must manually allocate those additional units by updating the rules.

Updating MAC Address

AppDynamics licenses for on-premises Controllers are tied to the MAC address of the machine on which you installed the Controller.
The original MAC address set up must be done by AppDynamics Operations when a new on-premises license is first provisioned. From that point on, you may update your MAC address in a self-serve manner.

You can change the MAC address up to three times within a 12-month period following the below steps, after which, you will need to contact your Account Owner if you have to change it again.

To update the MAC address on a permanent license:

2. Log in with your AppDynamics user credentials.
3. Ensure that your account is set up with ‘primary user’ privileges for the license.

Only users with primary user privileges for a permanent on-premises license can update MAC addresses. You can see who the primary user is by navigating to the Subscriptions page under Account. Only one user can have primary user privilege per license.

If the wrong person in your company is set up as the primary user, they must contact AppDynamics Support with the license name and the email address of the person who should be the primary user.

4. On the main navigation bar, click Account > Subscriptions.
5. Click Edit next to the MAC address you would like to update. The Edit Mac Address window opens.
6. Change the MAC address and click Save.
License Entitlements and Restrictions

On this page:
- Entitlements
- Definitions

License restrictions applicable to all products:
- License restrictions with respect to any third party components included in the AppDynamics software, with which customers are required to comply, are located on the Legal Notices page.
- Compatibility limitations with respect to any third party components are located on the Supported Environments page.
- Customers are prohibited from using the MySQL database(s) included with the AppDynamics software with any other product or for any other purpose other than for the AppDynamics software as provided.
- Each unit of the products in the Application Performance Management section below includes one unit of Machine Agent and Universal Agent at no cost, but such Machine Agent and Universal Agent shall only be used on the instance of the OS being monitored by the purchased unit.
- Metric Data retention policy: Metrics in one-minute increments are retained for 4 hours; metrics in ten-minute increments are retained for 48 hours; metrics in one-hour increments are retained for 365 days.
- Event Data retention policy: As described below, except that retention for on-premises deployments is configurable.
- Peak Edition - AppDynamics for Java, .NET or Node.js: During the License Term of the Subscription Licenses (or, if applicable, during the Maintenance and Support Term of perpetual licenses) on the applicable Order Form, End User may, once per calendar quarter and upon 24 business hours’ written notice, exchange licenses among the following types of Software units: AppDynamics Pro Edition for Java, .NET and Node.js on a one-for-one basis, and separately AppDynamics Test & Dev Edition (each an “Exchangeable Group”) for the same products on a one-for-one basis. For clarity, the list prices of all products within each Exchangeable Group (Pro Edition and Test & Dev Edition, respectively) are the same and through such exchanges, End User may not at any time use products with cumulative value exceeding the total list price of products on the applicable Order Form.
- Peak Edition - Browser or Mobile RUM: During the License Term of the Subscription Licenses (or, if applicable, during the Maintenance and Support Term of perpetual licenses) on the applicable Order Form, End User may, once per calendar quarter and upon 24 business hours’ written notice, exchange licenses among the following types of Software units: AppDynamics Pro Edition for Browser Real User Monitoring and Browser Analytics for units of Mobile Real User Monitoring and Mobile Analytics on a one-for-one basis (and vice versa), and separately AppDynamics Test & Dev Edition for the same products (and vice versa) on a one-for-one basis. End User may not exchange any AppDynamics for Browser Real User Monitoring licenses during any year in which End User has used any pageviews of such licenses. For clarity, the list prices of all products within each Exchangeable Group (Pro Edition and Test & Dev Edition, respectively) are the same and through such exchanges, End User may not at any time use products with cumulative value exceeding the total list price of products on the applicable Order Form.
- APM Any Language - APM Any Language customers are prohibited from instrumenting APM Any licenses on SAP application servers unless such customer has purchased the requisite AppDynamics for SAP - ABAP Agent(s).
- Upgrades - Upgrade entitle the customer to the entitlement of the product to which the customer is upgrading.
- Standard University Subscription - Customers that have purchased any of the software listed below under the heading “Application Performance Management” shall be entitled to access the AppDynamics University Self-Paced Library, subject to the conditions set forth in this paragraph. No greater than 20 unique users per month per AppDynamics account are entitled to this access. Access shall be limited to customers that have purchased this software either (i) on a subscription basis but remain current on payment of all subscription license fees, if any or (ii) on a perpetual basis but remain current on payment of all maintenance and support fees, if any. Customers may log in using AppDynamics account credentials.

Entitlements

With respect to the AppDynamics software, each license unit on an order document entitles the customer to the following:

<table>
<thead>
<tr>
<th>Product</th>
<th>Entitlement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peak</td>
<td></td>
</tr>
</tbody>
</table>
### Peak Edition - APM Any Language / AppDynamics for Java, .NET or Node.js (SaaS)
- One unit of APM Any Language / AppDynamics for Java, .NET or Node.js (SaaS); and
- One unit of Transaction Analytics (SaaS); and
- One unit of Network Visibility; and
- One unit of Server Visibility

### Peak Edition - APM Any Language / AppDynamics for Java, .NET or Node.js (on-prem)
- One unit of APM Any Language / AppDynamics for Java, .NET or Node.js (on-prem); and
- One unit of Transaction Analytics (on-prem); and
- One unit of Network Visibility; and
- One unit of Server Visibility

### Peak Edition - Browser or Mobile RUM (SaaS)
- One unit of Browser Real User Monitoring (SaaS) and one unit of Browser Analytics (SaaS); or
- One unit of Mobile Real User Monitoring (SaaS) and one unit of Mobile Analytics (SaaS)

### Peak Edition - Browser or Mobile RUM (on-prem)
- One unit of Browser Real User Monitoring (on-prem) and one unit of Browser Analytics (on-prem); or
- One unit of Mobile Real User Monitoring (on-prem) and one unit of Mobile Analytics (on-prem)

### Peak Edition - APM Any Language Microservices (SaaS)
- The same entitlement as 5 APM Any Language (SaaS) units, restricted to use with the following: Docker Containers, all CloudFoundry-based providers, Redhat OpenShift, Heroku Dyno, Microsoft Azure App Services (including Azure WebApps, Azure WebJobs and Azure API Apps), Microsoft Azure Service Fabric, Microsoft Azure Containers, Amazon Elastic Beanstalk, Oracle PaaS (Java and Node.js only) and Bluemix Containers; and
- One unit of Transaction Analytics (SaaS); and
- One unit of Network Visibility; and
- One unit of Server Visibility

### Peak Edition - APM Any Language Microservices (on-prem)
- The same entitlement as 5 APM Any Language (on-prem) units, restricted to use with the following: Docker Containers, all CloudFoundry-based providers, Redhat OpenShift, Heroku Dyno, Microsoft Azure App Services (including Azure WebApps, Azure WebJobs and Azure API Apps), Microsoft Azure Service Fabric, Microsoft Azure Containers, Amazon Elastic Beanstalk, Oracle PaaS (Java and Node.js only) and Bluemix Containers; and
- One unit of Transaction Analytics (on-prem); and
- One unit of Network Visibility; and
- One unit of Server Visibility

### Advanced Edition - APM Any Language
- One unit of APM Any Language (on-prem or SaaS as applicable); and
- One unit of Network Visibility; and
- One unit of Server Visibility

### Advanced Edition - APM Any Language Microservices
- The same entitlement as 5 APM Any Language (on-prem or SaaS as applicable) units, restricted to use with the following: Docker Containers, all CloudFoundry-based providers, Redhat OpenShift, Heroku Dyno, Microsoft Azure App Services (including Azure WebApps, Azure WebJobs and Azure API Apps), Microsoft Azure Service Fabric, Microsoft Azure Containers, Amazon Elastic Beanstalk, Oracle PaaS (Java and Node.js only) and Bluemix Containers; and
- One unit of Network Visibility; and
- One unit of Server Visibility

### Application Performance Management

<table>
<thead>
<tr>
<th>APM Any Language</th>
<th>One license unit from the following product types: AppDynamics for Java, .NET, Node.js, PHP, Python, Web Server, C++, Go, the rights to monitor 16 Value Units of IBM Mainframe capacity, or IBM Integration Bus.</th>
</tr>
</thead>
<tbody>
<tr>
<td>APM Any Language Microservices (5pack)</td>
<td>The same entitlement as 5 APM Any Language units, restricted to use with the following: Docker Containers, all CloudFoundry-based providers, Redhat OpenShift, Heroku Dyno, Microsoft Azure App Services (including Azure WebApps, Azure WebJobs and Azure API Apps), Microsoft Azure Service Fabric, Microsoft Azure Containers, Amazon Elastic Beanstalk, Oracle PaaS (Java and Node.js only) and Bluemix Containers.</td>
</tr>
<tr>
<td>Product</td>
<td>Entitlement Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>AppDynamics for Java</td>
<td>Instrument a single JVM instance; snapshots are retained for 14 days unless archived</td>
</tr>
</tbody>
</table>
| AppDynamics for Java Microservices | Instrument a single JVM instance with a heap allocation less than or equal to 1 GB  
Note: This entitlement shall only apply to existing customers of AppDynamics for Java Microservices that have licensed a SKU for AppDynamics for Java Microservices. This entitlement is mutually exclusive with the entitlement to use AppDynamics for Any Language licenses. |
| AppDynamics for .NET          | Instrument an unlimited number of CLRs on a single Windows OS instance; snapshots are retained for 14 days unless archived  
For APM Any Language Microservices, instrument an unlimited number of CLRs on a single instance of an application  
For .NET on Microsoft Azure for Cloud Services, instrument an unlimited number of CLRs running in a single instance of a Microsoft Azure Web role or Worker role |
<p>| AppDynamics for PHP           | Instrument a single PHP runtime instance; snapshots are retained for 14 days unless archived                                                                                                                                                                                                                                                                                                                                                                                                               |
| AppDynamics for Node.js       | Instrument up to 10 Node.js processes per OS instance; snapshots are retained for 14 days unless archived                                                                                                                                                                                                                                                                                                                                                                                                                  |
| AppDynamics for Python        | Instrument an unlimited number of Python processes running on a single OS instance; snapshots are retained for 14 days unless archived                                                                                                                                                                                                                                                                                                                                                                  |
| AppDynamics for Web Server    | Instrument an unlimited number of Apache Web Server instances on a single OS instance; snapshots are retained for 14 days unless archived                                                                                                                                                                                                                                                                                                                                                                                                |
| AppDynamics for C++           | Instrument an unlimited number of C++ runtime applications on up to 3 OS instances using the software development kit; snapshots are retained for 14 days unless archived                                                                                                                                                                                                                                                                                                                                             |
| AppDynamics for Go            | Instrument up to 3 Go processes; snapshots are retained for 14 days unless archived. This is available as part of the APM Any Language license only.                                                                                                                                                                                                                                                                                                                                                              |
| AppDynamics for SAP - ABAP Agent | Instrument a single application server                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| AppDynamics for IBM Integration Bus | Instrument an unlimited number of IBM Integration Bus processes on 1 OS instance. This is available as part of the APM Any Language license only.                                                                                                                                                                                                                                                                                                                                                           |
| Universal Agent               | Deploy on a single instance of an OS                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <strong>End User Monitoring</strong>       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| Browser Real User Monitoring (SaaS) | Instrument 10 Million Pageviews per 12-month period and access the AppDynamics-hosted EUM Cloud and AppDynamics-hosted Events Service; raw page requests and sessions are retained as Event Data. Customers may also choose to store raw Ajax or other events requests as Event Data, and every 5 requests stored consumes one Pageview. Event Data is retained for 8 days. Additional retention of Event Data for 30, 60 or 90 days is available as an add-on at additional cost, and requires the purchase of AppDynamics Browser Analytics (SaaS). |
| Browser Real User Monitoring (on-prem) | Instrument 10 Million Pageviews per 12-month period; raw page requests and sessions are retained as Event Data. Customers may also choose to store raw Ajax or other events requests as Event Data, and every 5 requests stored consumes one Pageview. Event Data is retained for 8 days by default. Customer is not entitled to access the AppDynamics-hosted EUM Cloud or AppDynamics-hosted Events Service. |
| Mobile Real User Monitoring (SaaS) | Instrument 5,000 MAAs per calendar month and access the AppDynamics-hosted EUM Cloud and AppDynamics-hosted Events Service. In addition, for each MAA unit, 500 events per day of the following types are stored on the event service as Event Data: network requests, sessions, breadcrumbs and custom data. Crash reports are retained for 365 days and all other Event Data is retained for 8 days. Additional retention of Event Data for 30, 60 or 90 days is available as an add-on at additional cost, and requires the purchase of AppDynamics Mobile Analytics (SaaS). |
| Mobile Real User Monitoring (on-prem) | Instrument 5,000 MAAs per calendar month. In addition, for each MAA unit, 500 events per day of the following types are stored on the event service as Event Data: network requests, sessions, breadcrumbs and custom data. Crash reports are retained for 365 days and all other Event Data is retained for 8 days by default. Customer is not entitled to access the AppDynamics-hosted EUM Cloud or AppDynamics-hosted Events Service. |</p>
<table>
<thead>
<tr>
<th>Plan</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Browser Synthetic Monitoring - Hosted Agent (SaaS)</td>
<td>Consumes 40,000 units of 5-second blocks of time on AppDynamics’ hosted synthetic network per calendar month, and use Business Performance Monitoring features on the synthetic Events Data, if applicable, which is retained for 13 months. Each run of a synthetic job is rounded up to the next 5 seconds. Unused time is not rolled over to the following month. Customer is entitled to access the AppDynamics-hosted EUM Cloud or AppDynamics-hosted Events Service, both of which are required to use this product.</td>
</tr>
<tr>
<td>Browser Synthetic Monitoring - Private Agent - Per Location (SaaS)</td>
<td>Run a synthetic job that hits 1 page from 1 location (multiple units add together to run jobs that hit multiple pages from multiple locations), and use Business Performance Monitoring features on the synthetic Events Data, if applicable, which is retained for 13 months. A job that fails to hit any pages is counted as 1 page. Customer is entitled to access the AppDynamics-hosted EUM Cloud or AppDynamics-hosted Events Service, both of which are required to use this product.</td>
</tr>
<tr>
<td>Browser Synthetic Monitoring - Private Agent - Unlimited Locations (SaaS)</td>
<td>Run a synthetic job that hits 1 page from any number of locations, and use Business Performance Monitoring features on the synthetic Events Data, if applicable, which are retained for 13 months. Customer is limited to 500,000 Pageviews per calendar month period. Unused Pageviews are not rolled over to the following month. A job that fails to hit any pages is counted as 1 page. Customer is entitled to access the AppDynamics-hosted EUM Cloud or AppDynamics-hosted Events Service, both of which are required to use this product.</td>
</tr>
<tr>
<td>Browser Synthetic Monitoring - Private Agent - Per Location (on-prem)</td>
<td>Run a synthetic job that hits 1 page from 1 location (multiple units add together to run jobs that hit multiple pages from multiple locations), and use Business Performance Monitoring features on the synthetic Events Data, if applicable, which are retained for 13 months. A job that fails to hit any pages is counted as 1 page. Customer is not entitled to access the AppDynamics-hosted EUM Cloud or AppDynamics-hosted Events Service.</td>
</tr>
<tr>
<td>Browser Synthetic Monitoring - Private Agent - Unlimited Locations (on-prem)</td>
<td>Run a synthetic job that hits 1 page from any number of locations, and use Business Performance Monitoring features on the synthetic Events Data, if applicable, which are retained for 13 months. Customer is limited to 500,000 Pageviews per calendar month period. Unused Pageviews are not rolled over to the following month. A job that fails to hit any pages is counted as 1 page. Customer is not entitled to access the AppDynamics-hosted EUM Cloud or AppDynamics-hosted Events Service.</td>
</tr>
<tr>
<td>IoT for Connected Devices (SaaS)</td>
<td>Instrument 1,000 application instances that are embedded on a connected device (or multiple devices) per calendar month and access to the AppDynamics-hosted Events Service, with a data maximum of 50 GB per account per day. Only customers with at least 100,000 application instances that are (i) embedded on a connected device (or multiple devices), (ii) eligible to be instrumented AppDynamics for IoT and (iii) are actively in use by customer’s end users shall be entitled to use AppDynamics for IoT. AppDynamics for IoT is only available to AppDynamics SaaS customers. An “eligible” application instance is an application that can be instrumented using AppDynamics’ C/C++ SDK, Java SDK or AppDynamics Rest APIs (each for IoT). Event Data is retained for 8 days. Additional retention of Event Data for 30, 60 or 90 days is available as an add-on at additional cost.</td>
</tr>
</tbody>
</table>

**Business Performance Monitoring**

<table>
<thead>
<tr>
<th>Plan</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transaction Analytics (SaaS)</td>
<td>Instrument up to 1,000,000 business transaction events per 24-hour period and access to the AppDynamics-hosted Events Service, with a data maximum of 50 GB per account per day for the following product types: AppDynamics for Java, .NET, and Node.js. Data retention limited to 8 days. Additional retention of 30, 60, or 90 days available as an add-on.</td>
</tr>
<tr>
<td>Transaction Analytics (on-prem)</td>
<td>Instrument up to 1,000,000 business transaction events per 24-hour period (limited to 90 days of data storage) per license unit. Customer is not entitled to access the AppDynamics-hosted Events Service.</td>
</tr>
<tr>
<td>Browser Analytics (SaaS)</td>
<td>Instrument 10 Million Pageviews per 12-month period and access the AppDynamics-hosted EUM Cloud and AppDynamics-hosted Events Service; raw page requests and sessions are retained as Event Data with a default limit of 50 GB per account per day for each Event Data type. Customers may also choose to store raw Ajax or other events requests as Event Data, and every 5 requests stored consumes one Pageview. Event Data is retained for 8 days. Additional retention of Event Data for 30, 60 or 90 days is available as an add-on at additional cost.</td>
</tr>
<tr>
<td>Browser Analytics (on-prem)</td>
<td>Instrument 10 Million Pageviews per 12-month period; raw page requests and sessions are retained as Event Data with a default limit of 50 GB per account per day for each Event Data type. Customers may also choose to store raw Ajax or other events requests as Event Data, and every 5 requests stored consumes one Pageview. Event Data is retained for 8 days by default. Customer is not entitled to access the AppDynamics-hosted EUM Cloud or AppDynamics-hosted Events Service.</td>
</tr>
</tbody>
</table>
Mobile Analytics (SaaS) Instrument 5,000 MAAs per calendar month and access the AppDynamics-hosted EUM Cloud and AppDynamics-hosted Events Service. In addition, for each MAA unit, 500 events per day of the following types are stored on the event service as Event Data: network requests, sessions, breadcrumbs and custom data with a default limit of 50 GB per account per day for each Event Data type. Crash reports are retained for 365 days and all other Event Data is retained for 8 days. Additional retention of Event Data for 30, 60 or 90 days is available as an add-on at additional cost.

Mobile Analytics (on-prem) Instrument 5,000 MAAs per calendar month. In addition, for each MAA unit, 500 events per day of the following types are stored on the event service as Event Data: network requests, sessions, breadcrumbs and custom data with a default limit of 50 GB per account per day for each Event Data type. Crash reports are retained for 365 days and all other Event Data is retained for 8 days by default. Customer is not entitled to access the AppDynamics-hosted EUM Cloud or AppDynamics-hosted Events Service.

Log Analytics (SaaS) Publish/index 5 GB of log data per 24-hour period (limited to 8 days of data storage) and access the AppDynamics-hosted Events Service. Additional retention beyond 8 days can be purchased as an add-on.

Log Analytics (on-prem) Publish/index 5 GB of log data per 24-hour period (limited to 90 days of data storage). Customer is not entitled to access the AppDynamics-hosted Events Service.

Infrastructure Visibility

<table>
<thead>
<tr>
<th>Machine Agent</th>
<th>Instrument a single instance of an OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server Visibility</td>
<td>Instrument a single instance of an OS</td>
</tr>
<tr>
<td>Network Visibility</td>
<td>Instrument an unlimited number of Java nodes on a single instance of an OS</td>
</tr>
<tr>
<td>Service Availability Monitoring</td>
<td>Instrument 60 HTTP based services</td>
</tr>
</tbody>
</table>

AppDynamics for Database Monitor one instance for the following databases: PostgreSQL, Microsoft SQL Server, MySQL, Sybase ASE, Sybase IQ, Couchbase Server, Oracle, IBM DB2.

Monitor one Microsoft Azure SQL database or one Microsoft Azure SQL Managed Instance.

Monitor one MongoDB replica set (or, if no replica sets exist, one standalone MongoDB instance) for MongoDB databases.

Event Data is retained for 10 days for each of the databases monitored by AppDynamics for Database.

AppDynamics for NetApp (on-prem) Monitor one NetApp controller for the following NetApp controller: FAS2nnn and FAS3nnn. Event Data is retained for 14 days by default.

Monitor ½ NetApp controller for the following NetApp controllers: FAS6nnn and FAS8nnn series (two license units are required per NetApp controller). Event Data is retained for 14 days by default.

Definitions

- "CLR" means Common Language Runtimes
- "EUM Cloud" means the end user monitoring component that receives and processes data received from end user devices
- "Dedicated Controller" means a version of the central repository and analytics engine component of the Software where End User’s data is logically separated from that of other customers
- "Event Data" means the raw data generated by the Software and stored in the Events Service
- "Events Service" means the component that stores the Event Data
- "GB" means gigabyte
- "Instrument" means to collect data, monitor, troubleshoot and manage using the AppDynamics software
- "JVM" means Java Virtual Machine
- "Metric Data" means the aggregate value of the data collected by the Software over a given period of time
- "Monthly Active Agent" or "MAA" means a unique installation of an application instrumented with an AppDynamics Mobile Real User Monitoring Agent that was launched on a device in a given calendar month
- "OS" means an operating system
- "Pageview" means a request to load a single web page, virtual page or view of an Internet site
The Universal Agent provides a central management point for administering runtime agents for diverse application environments and eases the management of large environments. The Universal Agent automates the tasks of installing, running, and maintaining AppDynamics runtime agents, such as the Machine Agent and Java Agent.

After you setup the Universal Agent, you can update the versions of the runtime agents by putting new distributions in a central repository and updating the configuration with the latest version. To further ease agent management in large and dynamic environments, the Universal Agent configuration supports remote JVM attachment and dynamic node naming. This topic describes how to set up and use the Universal Agent.

### About the Universal Agent

The Universal Agent enables you to run and manage AppDynamics runtime agents by using a single configuration file, called a rulebook. The rulebook determines whether the Universal Agent installs, starts, and stops runtime agents. By default, the Universal Agent runs in controller mode and applies a shared rulebook stored and served by the Controller. Alternatively, you can run the Universal Agent in local mode, in which it uses a rulebook (stored in the Universal Agent home directory) that is specific to the instance of the agent. See Universal Agent Rulebooks for more details on operation modes.

When the Universal Agent finds a new rule for a runtime agent in the rulebook, it retrieves the runtime agent from a shared repository, installs the agent as a local monitor, and starts the agent. You can manage multiple runtime agents on a single host, which allows for having different versions of the runtime agents for each JVM or each host in your environment.

### Set Up Tasks

To use the Universal Agent, you need to do the following tasks:
1. **Set up** the runtime agent source repository as either a Controller repository or file system repository. A file system repository would typically be on a shared repository accessible to the Universal Agent host machines.
2. **Install** the Universal Agent on the machines to be monitored.
3. **Create** the rulebooks to manage your runtime agent instances.

### Getting Started

The following procedures walk you through the three set-up tasks for a simple scenario using a single Linux-based application server with an on-premises Controller. On Linux, these steps require `sudo` access to the application server machine, because the setup establishes the Universal Agent as a service.

When following these steps, adjust the values in the commands for your specific Controller, user, account, and so on. Before starting, review additional information in **Install the Universal Agent**.

### Set up the runtime agent source repository

This procedure sets up a Controller source repository.

1. Confirm that you have an on-premises Controller installed and running and you have access to the Controller machine.
2. On the controller, create the `<controller-home>/agent-binaries` directory. This directory serves as your Controller source repository for the runtime agent binaries.
3. Put the Universal Agent ZIP file in the directory.
4. Rename it to `universalagent-<version>-64bit-linux.zip`.
5. Download the Machine Agent ZIP file to the Controller source repository, `<controller-home>/agent_binaries`.
6. Rename the Machine Agent file to `machine-<version>-64bit-linux.zip`. (Note there is no bitness in this name.)
7. Download the Java Agent ZIP to the Controller source repository, `<controller-home>/agent_binaries`.
8. Rename the Java Agent file to `java-<version>.zip`. (Note there is no bitness in this name.)
9. Continue to the next section to install the Universal Agent.

### Install the Universal Agent on the machines to be monitored

1. Get the account key for your Controller:
   a. From the Controller UI, open the License page and click the **Accounts** tab.
   b. Click the link to show the key.
2. From the application server machine, get the Universal Agent distribution. You can use the following CURL command to get the Universal Agent from the Controller repository. Replace the `<controller_user>`, `<controller_account>`, `<controller_password>`, and `<controller_host>` address and `<primary_port>` with values for your system. The Controller user should be an account in the Controller UI that has Universal Agent administration permissions. Replace `<account_access_key>` with your key and use the version of your specific Universal Agent.

```bash
curl -u '<controller-user>:@<controller-account>:<controller-password>':
-X POST -d
'action=downloadAgent&agentVersion=<version>&agentName=universalagent&classifier=64bit-linux' -o ua<version>.zip
'http://<controller-host:primary-port>/controller/DownloadServlet'
mkdir ua_install
unzip ua<version>.zip -d ua_install/
sudo ./ua-install/ua<version>/bin/install.sh
--account-access_key <account-access_key>
```
By default, the script installs the agent in the directory, `/opt/appdynamics/universal-agent/`. You can change that using the argument `--target`, but the steps (below) for creating your rulebooks assume the default installation location, `/opt`.

Create the rulebook

1. From the application server machine, where you installed the Universal Agent, execute a REST API command against your Controller to add a single default rulebook.
   Replace `<controller-user>`, `<controller-account>`, `<controller-password>`, `<controller-host>`, and `<controller-port>` with the values for your Controller.
curl -i -X PUT -su
'<controller-user>@<controller-account>:<controller-password>'
-H "Content-type: application/json" -H
"Accept:application/json"

"http://<controller-host>:<controller-port>/controller/universalagent/v1/user/rulebooks/byName/default-controller" --data '{
 "name": "default-controller",
 "comments": "An example rule book to monitor Java Agent, Machine Agent and Universal Agent itself.",
 "rules": [

  {
   "config": {
     "state": "started",
     "version": "<version>"
   },
   "monitor": "machine",
   "comments": "Rule to monitor Machine Agent",
   "condition": "True",
   "name": "Machine monitor"
  },

  {
   "name": "Universal Agent rule",
   "comments": "Rule to monitor Universal Agent",
   "monitor": "universal",
   "config": {
     "version": "<version>",
     "state": "started"
   },
   "condition": "True"
  },

  {
   "config": {
     "version": "<version>",
     "state": "started",
     "application_name": "YourAppName",
     "tier_name": "1stTier"
   },
   "condition": "True"
   "name": "Java Monitor",
   "comments": "Rule to monitor Java Agent",
   "monitor": "java"
  }
]
}'
2. After a few moments, verify that the Universal Agent retrieves and installs the Machine Agent and the Java Agent. The corresponding agent files should now appear in the following directories:
   - Machine Agent: /opt/appdynamics/universal-agent/monitor/machine
   - Java Agent: /opt/appdynamics/universal-agent/monitor/java
   - or <target_dir>/appdynamics/universal-agent/monitor/... if you specified --target <target_dir> in the installation step

3. Check the status of the Universal Agent as reported to the Controller using the REST API, universalagent/v1/user/agents/summary. For instance, using cURL, enter the following command: Replace the username, password, and Controller address with values appropriate for your environment. The response should list the running agents with the version number and the applied rulebook for each.

```bash
curl -s -X GET -u
'<controller-user>@<controller-account>:<controller-password>'
-H 'Content-type: application/json'
http://<controller-host>:<controller-port>/controller/universalagent/v1/user/agents/summary
```
Install the Universal Agent

The AppDynamics Universal Agent runs on each app server machine where you want to deploy runtime agents. The first step in using the Universal Agent to manage your agent deployment, therefore, is installing the Universal Agent on the monitored machines.

This topic introduces Universal Agent installation. When you are ready to install the Universal Agent, see the topic specific to your operating system:

- Install the Universal Agent on Windows
- Install the Universal Agent on Linux

Supported Environments and Requirements

The Universal Agent occupies about 20 MB of disk space. When you start the installation, the script first checks the system for sufficient space. Using the Universal Agent to add more runtime agents to your target machine later will require additional space.

The installation process installs the Universal Agent as an automatically started system service. Therefore, you need to install on the system as a user with sufficient privileges for this type of installation. On Linux, for example, you typically need to run the script as a user with sudo privileges. See Permissions for Running the Universal Agent.

The Universal Agent can support the deployment and management of the following runtime agents:

- Standalone Machine Agent
- Java Agent
- .NET Agent (Windows only)
- Analytics Agent
- Network Agent (Linux only)

Supported environments and requirements for the runtime agents still apply even when the Universal Agent is managing them. For example, in 4.4, the Network Agent is only available on Linux. Be sure to review the requirements for each runtime agent.

Some Universal Agent features (such as automatic JVM attachment and dynamic rulebook value propagation) have additional requirements and limitations, which are discussed in separate topics.

The Universal Agent is available on the following Linux and Windows versions:

Windows

Installing the Universal Agent on Windows requires the Universal C Runtime. If it the Universal C Runtime is not installed on your Windows server you should install the Update for Universal C Runtime in Windows.

AppDynamics supports the Universal Agent on:

- Windows Server 2008 SP2 and higher

Linux

Installing the Universal Agent on Linux requires the GNU C Library version 2.12, commonly known as glibc 2.12. The minimum versions of the common distributions that meet this requirement are:

- Ubuntu 11.04+
- CentOS 6+
- Red Hat Enterprise Linux 6+
Controller Connection Settings

The initial configuration for the Universal Agent specifies the Controller host, port, account name, and account key, so you will need to know the values to use before starting. These settings are equivalent to those used by other types of agent. You can find out more in the Agent-to-Controller Connections topic. Note that, on Linux, if you acquire the Universal Agent from the Controller, the settings are preconfigured.

Maintaining Universal Agents

After initial installation, you can update new versions of the Universal Agent itself using the Universal Agent rulebook. For more information, see Universal Agent Rules.
Install the Universal Agent on Windows

To use the AppDynamics Universal Agent, you install it on each machine where you want to use it to manage runtime agents, as described here.

When invoked, the installer checks your system for some requirements, such as sufficient disk space. When done, it leaves the agent running and installed as a Windows service.

**Install the Universal Agent on Microsoft Windows**

The two files on the download site for Microsoft Windows installations are MSI and ZIP. The ZIP file is designed to be uploaded to the repository and used for Universal Agent self-upgrade via rulebook rules. To install the Universal Agent on Windows, use the MSI installer.

1. **Prepare to install:** Installing the Universal Agent on Windows requires the Universal C Runtime. If you have not installed the Universal C Runtime on your Windows server, you should install the Update for Universal C Runtime in Windows.
2. **Download the MSI installation package from the AppDynamics download site.**
3. **Start the installation from the command line as follows.** For your package, you must use the path of your MSI file and specify a location for the installation log.

   ```
   msiexec /i "C:\MyPackage\Example.msi" /L*V "C:\log\example.log"
   ```

   The installer starts and the End User Agreement appears.
4. As you review the license information, scroll to the end of the agreement, and then click **Next** to continue the installation.
5. **Select the checkbox that indicates your acceptance of the agreement, and then click Next to continue the installation.**
6. **In the Configure pane, enter the following information and then click Next:**
   a. **Controller host:** The hostname or IP address for the Controller.
   b. **Controller port:** The primary listening port for the Controller. If not specified, the Universal Agent uses port 80 or 443 (with SSL enabled), by default.
   c. **Account name:** The name and access key for the Controller account where the agents should report data. You can get the Account Name and Access Key value from the **License Management** page in the Controller UI.
   d. **Account access key:** The name and access key for the Controller account where the agents should report data. You can get the Account Name and Access Key value from the **License Management** page in the Controller UI.
7. **Confirm the destination directory for the Universal Agent runtime files and local data and click Install.**
7. When the installation is complete, click **Finish**.

When done, the Universal Agent is running and installed as an automatically starting service.

**Windows Installer Command Line Arguments**

Instead of using the GUI installer, you can pass Universal Agent configuration settings to the MSI program from the command line in the following format:

```
msiexec.exe /i <name_of_UA_package>.msi /qn [ CONTROLLERHOST=<hostname> ] [ CONTROLLERPORT=<primaryport> ] [ ACCOUNTNAME=<accountname> ] [ ACCOUNTACCESSKEY=<accountkey> ] [ CONTROLLERSSLENABLED= ]
```

The /qn switch suppresses the UI.

The command arguments are:

- **CONTROLLERHOST=<<value>>**
- **CONTROLLERPORT=<<value>>**
- **ACCOUNTNAME=<<value>>**
- **ACCOUNTACCESSKEY=<<value>>**
- **CONTROLLERSSLENABLED=<<value>>**

**CONTROLLERHOST=<<value>>**

Required. AppDynamics Controller hostname.

**CONTROLLERPORT=<<value>>**

Required. AppDynamics controller port number.
ACCOUNTNAME=<value>
Required. Name of the account under which the Universal Agent will report to the controller. If you are running a single tenant controller, use the name of the default, built-in account, customer1.

ACCOUNTACCESSKEY=<value>
Required. Account access key for the Universal Agent to authenticate with the controller.

CONTROLLERSSLENABLED=<value>
Optional. Specifies whether or not to use SSL for the connection. If included in the command line and set to any value, the Controller uses SSL. If excluded from the command line, SSL is not required.

Example
The command line is in the following format:

```
msiexec.exe /i universalagent-setup-4.3.0.0-64bit-windows /qn
CONTROLLERHOST=controller.sample.host CONTROLLERPORT=8090
ACCOUNTNAME=customer1 ACCOUNTACCESSKEY=ABCd-10123-XYZD-0123
CONTROLLERSSLENABLED=1
```

Start and Stop the Universal Agent Windows Service

You can stop and start the Universal Agent from the Windows services manager. To do so, find the Universal Agent service and use the services manager controls to stop and start it.

Uninstall the Universal Agent

To uninstall the Universal Agent, use the Windows MSI program, as follows:

1. Disable auto-java (if you enable it) before uninstalling the Universal Agent. Use the --disable-auto-java command.
2. From the command terminal, enter the following command:

```
msiexec.exe /x <name_of_UA_package>.msi
```

3. Follow the MSI tool prompts to complete the uninstallation.
4. Reboot the machine after the uninstall is complete.
Install the Universal Agent on Linux

On this page:

- About the Linux Installation
- Install the Universal Agent on Linux
- Linux Install Script Format
- Start and Stop the Universal Agent Linux Service
- Uninstall the Universal Agent

This topic describes how to install the AppDynamics Universal Agent on Linux systems.

About the Linux Installation

To install the Universal Agent on Linux, you use the installation script, `install.sh`. The script accepts configuration parameters in various forms:

- As settings in the `conf/universalagent.yaml` file. (If you download the Universal Agent from the Controller, this file may be preconfigured for your environment.)
- As command-line arguments you enter when invoking the install script.
- As settings in a parameter file named by the `-p` argument.

If the installer does not find the Controller host or port settings in one of these sources, it prompts you for the values at the command line.

The following steps illustrate a simple installation scenario. The procedure assumes you already have an AppDynamics Controller installed and running. When installation is completed, the installation process leaves the Universal Agent running on the target machine.

Install the Universal Agent on Linux

1. Download the Universal Agent ZIP file to the machine where you want to deploy runtime agents.
   - To download from the controller source repository, use the following command. Substitute the placeholders with your username, account name and password, as indicated. Also replace `<controller_host>`, `<controller_port>`, `<username>` and `<password>` with your username, account name and password, as indicated. Also replace `<controller_host>` with your controller hostname or IP address and `<controller_port>` with the primary listening port for your Controller.
     
     ```bash
     curl -u <username>@<account_name>:<password> \
     -X POST \n     -d 
     'action=downloadAgent&agentVersion=4.4.0.0&agentName=universalagent&classifier=64bit-linux' \n     -o ua4.4.0.0.zip 
     'http://<controller_host>:<controller_port>/controller/DownloadServlet'
     ```

   - If you are not using the Controller repository, you can directly copy the Universal Agent binary onto the machine where you want to deploy runtime agents.

2. Extract the contents to the agent installation directory. This directory is referred to as `<universal_agent_home>` in these instructions.
3. Specify the initial configuration settings for the Universal Agent. For instance, to configure the settings in the Universal Agent configuration file, follow these steps:
   a. Navigate to the `conf` directory and open `universalagent.yaml` for editing. This file contains the configuration properties for the Universal Agent.
   b. Configure the settings with values that correspond to your Controller environment. The Universal Agent, and unless otherwise configured, the deployed runtime agents, use this information to connect to your Controller:
      - `controller_host`: The hostname or IP address for the Controller.
      - `controller_port`: The primary listening port for the Controller. If not specified, the Universal Agent uses port 80 or 443 (with SSL enabled), by default.
      - `account_name` and `account_access_key`: The name and access key for the Controller account where the agents should report data. You can get the Account Name and Access Key from the License Management page.
   c. If you want your agent to be registered with a given name, un-comment the `name` tag under the `agent` section and provide a value such as:

   ```yaml
   name: your_agent_name
   ```

   If you do not provide an agent name, the name of the agent defaults to the hostname of the machine.

   Note that `.yaml` files use a fixed indentation scheme. Therefore, be careful that all the sections are indented correctly. Make sure that the sections `controller` and `agent` are indented correctly with the right number of spaces (not tabs). For example, if you un-comment `account_name`, `account_access_key` under `controller` section, you need to add a space to make it align with the other tags.

   Do not use tabs in `universalagent.yaml`. The YAML loader used by the agent does not support tabs.

4. Specify the repository location. This is either the file location or network location where the Universal Agent gets the app agents to install. If using a local repository, see Runtime Agent Repository for instructions on setting up the repository.
5. Review and set any other properties in the configuration file applicable to your environment, as indicated by the inline comments.
6. Run the install script to install the Universal Agent into a Linux or Unix system.

   ```bash
   sudo ./ua4.5.0.0/bin/install.sh
   ```

   The Universal Agent starts immediately when installation is complete. It is registered as an automatically started service, so the Universal Agent starts automatically upon system reboots.

   **Linux Install Script Format**

   The syntax to run the install script is the following:
install.sh [ -p <param file>] [ --controller_host <controller host>] [ --controller_port <controller port>] [ --account_name <account name>] [ --account_access_key <account access key>] [ --target <target directory>]

The following sections describe each setting:

- **-p <param_file>**
  Optional. This optional argument specifies the full or partial path name of a text file that contains all of the arguments supported by the script.
  When specified, the file must contain the same keywords and values specified on the `install.sh` command. For readability, these can be contained on multiple lines. For example:

  ```
  Example param file
  ```

  ```
  --controller_host localhost
  --controller_port 8080
  ```

  A value specified on the command-line of `install.sh`, overrides a value specified in the file referenced by the `-p` argument.

  - **--controller_host <value>**
    Required. AppDynamics Controller host name.

  - **--controller_port <value>**
    Required. AppDynamics Controller port number.

  - **--account_name <value>**
    Required. Name of the account under which the Universal Agent will report to the Controller. If you are running a single-tenant Controller, use the name of the default, built-in account, `customer1`.

  - **--account_access_key <value>**
    Required. Account access key for the Universal Agent to authenticate with the Controller.

  - **--target <value>**
    Optional. Specifies the target directory where the Universal Agent is installed. Default is `/opt`.

  - **--no_service**
    Optional. Specifies that the ua daemon should not be defined as a service. In this case, you must manually start the ua daemon.

  Example
In this example, the Controller host and account access key are found in `paramfile.txt`.

```
ua4.5.0.0/bin/install.sh -p paramfile.txt --controller_port 8081
--account_name customer1
```

Sample contents of the `paramfile.txt` are:

```
--controller_host localhost --controller_port 8080
--account_access_key 'abcdef$ghi'
```

The Controller port value specified in this file is ignored, because `--controller_port 8081` is specified on the command invocation.

**Start and Stop the Universal Agent Linux Service**

When the install script completes successfully, the Universal Agent is defined as a system service. You can start and stop it using Linux service commands.

In a systemd environment, use the following commands to start, stop, and restart the service:

- `systemctl start appdynamics-universal-agent`
- `systemctl stop appdynamics-universal-agent`
- `systemctl restart appdynamics-universal-agent`

In a non-systemd environment, use the following commands:

- `service appdynamics-universal-agent start`
- `service appdynamics-universal-agent stop`
- `service appdynamics-universal-agent restart`

**Uninstall the Universal Agent**

To uninstall the agent, as a `sudo` user run the CLI command passing the uninstall switch, as follows:

```
./ua --uninstall
```
Permissions for Running the Universal Agent

On this page:

- Linux
- Setting up the Non-root User for Universal Agent
- Windows

Related pages:

- Install the Universal Agent

This topic describes the permissions needed to run the Universal Agent. When the Universal Agent installs and starts other runtime agents, it starts them using the same user as the Universal Agent itself. During installation, the default user for running the Universal Agent is set to root. You can create a non-root user, for example `<universal_agent_user>`, and assign the appropriate permissions to that user.

The installation process installs the Universal Agent as an automatically started system service. Therefore, you need to perform the installation on the system as a user with sufficient privileges for this type of installation. On Linux, for example, you typically need to run the script as a user with sudo privileges.

For all environments you can create a specific user with the necessary read/write/execute permissions for running the Universal Agent:

- All files in the `<universal-agent-home>` installation directory should be readable by the Universal Agent.
- The user that runs the Universal Agent must have write privileges to the logging output directory and to the `/conf` directory in the agent installation directory.
- The user that runs the Universal Agent must have write privileges to the `conf` and `logs` directories in the `<universal_agent_home>` directory.
- In addition, the user that runs the Universal Agent needs execute access as described below.

**Linux**

**SystemD**

- `systemctl stop`: Stops the Universal Agent service
- `systemctl restart`: Restarts the Universal Agent after upgrade
- `systemctl disable`: Uninstalls the Universal Agent service

**Non-SystemD**

- `service stop`: Stops the Universal Agent service
- `chkconfig --del`: Uninstalls the Universal Agent service
- `service restart`: Restarts Universal Agent after upgrade

**Other Commands**

- `java` - to start and stop standalone Analytics JVM (usually only on Windows)
- `java -version` - to determine version of Java
- `sudo -u <user-id> java .../javaagent.jar` - to remote attach to a JVM, if JVM is running with a different user id than the Universal Agent
- `java .../javaagent.jar` - to remote attach to a JVM, if JVM running with same user id as UA
- `machine-agent` - invokes machine-agent script to start machine agent
- `/opt/appdynamics/universal-agent/ua --daemon` - to start the Universal Agent daemon, when it is not defined as a Linux service

**Setting up the Non-root User for Universal Agent**

In most Linux installations, you can configure sudo ability for the Universal Agent by editing the `/etc/sudoers` file using `visudo`. The following steps provide an example of this configuration change:

1. Edit `/etc/sudoers` using the `visudo` command.
2. Find the line with "Defaults requiretty" and change it to "Defaults !requiretty".
3. Find the line with "rootALL=(ALL) ALL". After this line, add the line "<user_name> ALL=(ALL) ALL", where "<user>" is the user ID that the Universal Agent service is running under.
4. (For Java Agent Remote Attach) When deploying Java Agents into environments using remote attach, if the Universal Agent runs as root or as the same user that runs the JVMs to which you want to remotely attach, no additional user configuration is required. However, if the Universal Agent runs as a non-root user that is not the same user used to run the target JVM, then you need to authorize the Universal Agent user to use sudo privileges to enable the Universal Agent to retrieve environment variables used in dynamic variable binding.

At the end of the /etc/sudoers file, add the following line:

```
<ua_user> ALL = NOPASSWD: /opt/appdynamics/universal-agent/ua, /usr/bin/java
```

The value of `<ua_user>` is the user id that the Universal Agent service is running under. Note that `/usr/bin/java` represents the fully-qualified path name for Java on this system. This value can be found by entering the `which java` command, and may be different from `/usr/bin/java`.

5. (For deploying the Network Agent) Installing the Network Agent using the Universal Agent requires elevated privileges for some commands. At the end of the `/etc/sudoers` file, add the following line:

```
<ua_user> ALL = NOPASSWD: /bin/chmod, /bin/chown, /sbin/setcap
```

Note that `/sbin/setcap` represents the fully-qualified path name for `setcap` binary on this system. This value can be found by entering the `which setcap` command, and may be different from `/sbin/setcap`.

**Windows**

Windows permissions for files and subfolders are inherited by default from the parent folder (`<universal_agent_home>`). It is good practice to restrict permissions to users authorized to start, stop, and configure the Universal Agent:

- Read and Write permissions to all files and subfolders under `<universal-agent-home>`
- Permission to install and uninstall software
- Start, Stop, and Restart permissions for the Universal Agent service. You need admin privileges to install and run the service.
The Universal Agent can retrieve the runtime agent installation software from the Controller repository or from a local repository. This topic describes how to create each type of repository.

**Configure the Repository Location**

The repository types and locations are defined in the Universal Agent configuration file, `universalagent.yaml`, using the `repositories` keyword. You can specify a list of repository URLs (in order of preference) for retrieving AppDynamics monitor (agent) binaries and Universal Agent updates. The format is as follows:

```yaml
repositories:
  - file:///local/path
  - http://host:port/path
  - https://host:port/path
  - https://<username>:<password>@<host>:<port>/path
```

To configure a local repository, use the `file://` protocol prefix for the location. For example, if the local repository is in the `/users/appduser/repository` directory, configure the repository location as follows:

```yaml
repositories:
  - file:///Users/appduser/repository
```

Notice that there are three slashes after `file://`. The first two slashes indicate the file protocol, and the third slash refers to the root directory of the file system. All slashes should be forward slashes, even on Windows machines.

To configure a local repository on an HTTP server. Define the URL in any of the formats below:

```yaml
repositories:
  - https://host:port/path
  - http://host:port/path
```

You can also use basic authentication for an HTTP repository in the form:
repositories:
- https://<username>:<password>@<host>:<port>/<path>
- http://<username>:<password>@<host>:<port>/<path>

You can specify more than one repository location, in which case the Universal Agent uses the first agent file that matches a rule that it finds in the sequentially checked repositories. To use a combination of a local and Controller repository, include both, as follows:

repositories:
- file:///Users/appduser/repository
- https://<username>:<password>@<host>:<port>/<path>
- controller

The Universal Agent first checks the local repository and then the Controller for the runtime agent downloads.

Using a Controller Repository

This type of Universal Agent Repository is not available for SaaS Controllers. Use a local repository instead.

Downloading the .NET Agent from a Controller repository is not supported. Place the .NET Agent distribution file in a local repository.

Using the Controller as the repository location gives the Universal Agents in your environment a single, central point for accessing the runtime agents. To use the Controller repository, you need to be able to access the Controller installation directory.

1. Create a subdirectory named agent_binaries in the Controller home directory.
2. Download the runtime agent distribution files to the Controller home subdirectory agent_binaries from the download site.
3. Rename the downloaded agents to the format expected by the Universal Agent. Use the following naming conventions where <version> is the agent version number:
   - java-<version>.zip for non-IBM versions of the Java agent
   - java-<version>-ibm.zip for IBM versions of the Java agent
For example, java-4.5.0.0.zip or java-4.5.0.0-ibm.zip
   - universalagent-4.45-64bit-linux.zip
   - Your directory should look similar to the following:

   `<appdynamics_home>/
   Controller/
   agent_binaries/
   java-4.5.0.0.zip
   java-4.5.0.0-ibm.zip
   universalagent-4.45-64bit-linux.zip`

4. Specify the repository location to the Universal Agent, as described in Configure the Repository Location.

Using a Local Repository
If putting the repository in the Controller directory structure is not feasible, you can serve the agent distribution files from a local repository. The likely location for the repository, in this case, would be a shared network directory. The directory would need to be mounted by the machines running the Universal Agent.

The Universal Agent depends upon a defined directory structure within the repository directory and conventionally named runtime agent distribution files, as described in the following steps.

To create your own local Universal Agent repository:

1. Create the repository directory with the structure shown in the following example:

```
repository/
  monitor/
    java/
      4.5.0.0
        java-4.5.0.0.zip
        java-4.5.0.0.zip
    machine/
      4.5.0.0
        machine-4.5.0.0-64bit-linux.zip
```

Your local directory should similarly have a directory named `monitor` that contains subdirectories for the runtime agent types, including `java` and `machine` and their respective versions, as illustrated by the example.

2. Download the agent files to the appropriate location in the directory based on the agent type and version.

3. Rename the downloaded agents to the format expected by the Universal Agent. For example, rename `AppServerAgent-<version>.zip` to `java-<version>.zip`. See Agent Naming Format.

4. Specify the repository location to the Universal Agent, as described in Configure the Repository Location.

Agent Naming Format

The Universal Agent requires the downloaded runtime agent ZIP file in the repository to have a specific name format. When placing runtime agent files in the repository, ensure that they are named using the following format:

<table>
<thead>
<tr>
<th>Format</th>
<th>Examples</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>java-&lt;version&gt;.zip</td>
<td>java-4.5.0.0.zip</td>
<td>Java App Server Agent</td>
</tr>
<tr>
<td>machine-&lt;version&gt;-&lt;bitness&gt;&lt;bit&gt;-&lt;os&gt;.zip</td>
<td>machine-4.5.0.0-64bit-windows.zip</td>
<td>Machine Agent for 64-bit Windows</td>
</tr>
<tr>
<td></td>
<td>machine-4.5.0.0-32bit-windows.zip</td>
<td>Machine Agent for 32-bit Windows</td>
</tr>
<tr>
<td></td>
<td>machine-4.5.0.0-64bit-linux.zip</td>
<td>Machine Agent for 64-bit Linux</td>
</tr>
<tr>
<td></td>
<td>machine-4.5.0.0-32bit-linux.zip</td>
<td>Machine Agent for 32-bit Linux</td>
</tr>
<tr>
<td><strong>universalagent-&lt;version&gt;-&lt;bitness&gt;bit-&lt;os&gt;.zip</strong></td>
<td><strong>universalagent-4.5.0.0-64bit-windows.zip</strong></td>
<td><strong>Universal Agent for 64-bit Windows</strong></td>
</tr>
<tr>
<td>--------------------------------------------------</td>
<td>---------------------------------------------</td>
<td>---------------------------------------</td>
</tr>
<tr>
<td><strong>universalagent-4.5.0.0-32bit-windows.zip</strong></td>
<td><strong>Universal Agent for 32-bit Windows</strong></td>
<td></td>
</tr>
<tr>
<td><strong>universalagent-4.5.0.0-64bit-linux.zip</strong></td>
<td><strong>Universal Agent for 64-bit Linux</strong></td>
<td></td>
</tr>
<tr>
<td><strong>universalagent-4.5.0.0-32bit-linux.zip</strong></td>
<td><strong>Universal Agent for 32-bit Linux</strong></td>
<td></td>
</tr>
<tr>
<td><strong>analytics-agent-&lt;version&gt;.zip</strong></td>
<td><strong>analytics-agent-4.5.0.zip</strong></td>
<td><strong>Analytics Agent</strong></td>
</tr>
<tr>
<td><strong>analytics-agent-bundle-&lt;bitness&gt;bit-&lt;os&gt;-&lt;version&gt;.zip</strong></td>
<td><strong>analytics-agent-bundle-64bit-linux-4.5.0.0.zip</strong></td>
<td><strong>Bundled Analytics Agent</strong></td>
</tr>
<tr>
<td><strong>dotNetAgentSetup&lt;bitness&gt;-&lt;version&gt;.msi</strong></td>
<td><strong>dotNetAgentSetup64-4.5.0.0.msi</strong></td>
<td><strong>Windows .NET Agent</strong></td>
</tr>
<tr>
<td><strong>network-&lt;version&gt;-&lt;bitness&gt;bit-&lt;os&gt;.zip</strong></td>
<td><strong>network-4.5.0.0-64bit-linux.zip</strong></td>
<td><strong>Network Agent</strong></td>
</tr>
</tbody>
</table>

Notice that the Universal Agent and the Machine Agent include the bit number of the target operating system, because the downloads bundle the JRE.
Universal Agent CLI

On this page:

- Running the Universal Agent CLI Tool
- CLI Tool Command Syntax
- Command Line Arguments

You can perform administrative tasks and configuration changes for a Universal Agent using the Universal Agent command line interface (CLI), as described in this topic.

Running the Universal Agent CLI Tool

You can start the Universal Agent, or administer and configure a running Universal Agent, from the command line by invoking the `/opt/appdynamics/universal-agent/ua` program.

Configuration changes you make with the Universal Agent command line tool do not impact the content of the Controller rulebook. All rulebook changes affect only the local copy of the rulebook, `local.json`. Command line arguments that cause rulebook changes also direct the background Universal Agent daemon task to switch to Local mode, so that it operates from the `local.json` rulebook rather than the Controller rulebook.

CLI Tool Command Syntax

The following listing shows the CLI tool syntax:

```
/opt/appdynamics/universal-agent/ua [ --daemon] [ --start-rule <rule_names> ] [ --stop-rule <rule_names> ] [ --display-mode ] [ --set-mode [local | controller ] [ --modify-config <rule.attribute=value> ] [ --show-java-arguments <process_ids> ] [ --check-config ] [ --show-daemon-status ] [ --setup-win-service [ --data-dir <data_directory_path>] ] [ --restart-service ] [ --start-service ] [ --enable-auto-java ] [ --disable-auto-java ] [ --uninstall ]
```

You can pass more than one argument to the command. For example:

```
/opt/appdynamics/universal-agent/ua --start-rule rule1 rule2  --stop-rule rule3  --display-mode
```

Command Line Arguments

Quotes are necessary if the rule name contains spaces, otherwise quotes are optional. This is true for all commands that reference rule names.

- `--daemon`
- `--start-rule <rule_names>`
--stop-rule <rule_names>
--display-mode
--set-mode [ local | controller ]
--modify-config <rule_name> <attribute_name>=<new_value>
--show-java-arguments [ <process_ids> | all ]
--check-config
--show-daemon-status
--setup-win-service
--restart-service
--start-service
--enable-auto-java
--disable-auto-java
--enable-ldpreload
--disable-ldpreload
--uninstall

--daemon

Starts the Universal Agent as a daemon process. It is mutually exclusive with all other command line arguments.

--start-rule <rule_names>

Changes the state of one or more rules in the rulebook to a value of started. The rules to be modified are identified by their rule names. The use of the --start-rule argument switches the operation mode of the Universal Agent to Local. You can change the state of multiple rules within a single invocation. For example, /opt/appdynamics/universal-agent/ua --start-rule rule1 rule2 changes the state of both rule1 and rule2 to started.

The meaning of this state change varies from one monitoring agent to another.

- Machine Agent and Network Agent: A state value of started causes the Universal Agent background task to start the specified agent.
- Java Agent: A value of started activates the Java Agent within a matching JVM when that JVM starts. If the JVM has already started, it needs to be restarted for the Java Agent start to take effect.
- .NET Agent: Changing the state of the rule to started rules causes the specified rule to become `started` and installs and starts the .NET agent on the machine.
- Analytics Agent
  - For Linux platforms, the Analytics Agent runs as a Machine Agent extension. A value of started causes the Machine Agent to be configured to start the analytics extension the next time it is started.
  - For Windows platforms, the Analytics Agent runs in a standalone JVM. A value of started causes this JVM to be started if it is not already running.

--stop-rule <rule_names>

Changes the state of one or more rules in the rulebook to a value of installed. The rules to be modified are identified by their rule names. Using the --stop-rule argument switches the operation mode of the Universal Agent to Local. You can change the state of multiple rules within a single invocation. For example, /opt/appdynamics/universal-agent/ua --stop-rule rule1 rule2 changes the state of both rule1 and rule2 to installed.

The meaning of this state change varies from one monitoring agent to another.

- Machine Agent and Network Agent: A state value of installed causes the Universal Agent background task to stop the specified agent if it is running.
- Java Agent: The Universal Agent cannot stop a Java Agent because it runs as part of an application JVM. A state value of installed prevents the auto-java feature (if enabled) from activating the Java Agent within a matching application JVM.
- .NET Agent: Changing the state of the rule to installed stops the .NET agent and uninstalls it.
- Analytics Agent:
  - For Linux platforms, the Analytics Agent runs as a Machine Agent extension. A value of installed causes the Machine Agent to be configured to not start the analytics extension the next time it is started.
  - For Windows platforms, Analytics Agent runs in a standalone JVM. A value of installed causes this JVM to be stopped if it is currently running.

--display-mode
Displays the operation mode of the background Universal Agent daemon. It does not change anything.

```bash
--set-mode [ local | controller ]
```

Changes the operation mode of the background Universal Agent daemon to the specified mode.

If `local` is specified, and the daemon is currently in controller mode, then the current controller rulebook (as saved in the `controller-book.json` file) is renamed `local.json`.

If `controller` is specified, and the daemon is currently in local mode, then the `local.json` file is renamed `local.json.backup` (if possible).

```bash
--modify-config <rule_name> <attribute_name>=<new_value>
```

Changes an arbitrary attribute within the current rulebook to a new value. The use of the `--modify-config` argument switches the operation mode of the Universal Agent to Local.

You can specify the attribute in two ways:

- ```bash
    <rule_name> <attribute_name>=<new_value>
  ```
  this form changes a non-config attribute within the rule. `<rule_name>` identifies the name of the rule to be changed; `<attribute_name>` identifies the specific attribute; and `<new_value>` specifies the new value to be assigned to the attribute.
  For example: `/opt/appdynamics/universal-agent/ua --modify-config rule1 condition=False` changes the condition attribute of rule1 to a value of False.

- ```bash
    <rule_name> config.<attribute_name>=<new_value>
  ```
  this form changes a config attribute within the rule. A config attribute is a monitor-specific attribute that is defined within the "config" attribute set for the rule. `<rule_name>` identifies the name of the rule to be changed; `<attribute_name>` identifies the specific config attribute; and `<new_value>` specifies the new value to be assigned to the attribute. For example: `/opt/appdynamics/universal-agent/ua --modify-config rule1 config.state=start` has the effect of changing the `state` config attribute of rule1 to a value of started.

```bash
--show-java-arguments [ <process_ids> | all ]
```

Displays the changes that need to be made to a Java command line to correctly deploy the Java app agent. This command is helpful if you need to manually configure one or more Java startup scripts.

One or more arguments can be provided for this option. If a single `all` argument is provided, then this command displays the command line arguments that need to be defined for each JVM in the system that matches at least one of the Java rules. Otherwise, the arguments following `ua --show-java-arguments` should be the process ids for the processes that should be tested. If a process id represents a Java process, then the process is tested against the current Java monitoring rules. If the process matches at least one rule, the command line arguments that should be added to the process startup script are displayed.

Example:

```
Example of --show-java-arguments

```u
ua --show-java-arguments 123
Java arguments for pid 123
-javaagent:/opt/appdynamics/universal-agent/monitor/java/javaagent.jar
-Dappdynamics.agent.applicationName=MyApp,-Dappdynamics.agent.tierName=My Tier,-Dappdynamics.agent.reuse.nodeName=true,-Dappdynamics.ua.appagent.version=4.3.0.0
```
--check-config

Enables the Universal Agent to validity check the conf/universalagent.yaml configuration file for syntax errors. Reports if there are errors in the file that would cause the ua daemon process to fail initialization. The Universal Agent install script uses this argument to ensure a valid configuration file prior to installing the Universal Agent as a system service.

--show-daemon-status

Reports the status of the ua daemon process by scanning the processes running in the current OS and reporting the process ids of all the processes that are running the Universal Agent as a daemon. Run this command as the root user or by using sudo.

**Linux only:** When the Universal Agent daemon runs as a system service, it is normal for two process ids to be reported

--setup-win-service

Configures the Universal Agent as a service in the Windows environment. This command starts the Windows service (if it is not already running), and configures the registry so that the service starts automatically when the system is rebooted. The --data-dir command line option can be specified with --setup-win-service. The <data-directory-path> argument specifies the path name of the directory where the Universal Agent's data directories reside. These directories include log, conf, download, and rulebook. If the --data-dir option is not specified, then these directories must be located in the main installation directory of the Universal Agent.

--restart-service

Stops and starts the Windows Universal Agent service.

--start-service

Starts the Windows Universal Agent service, unless it is already started.

--enable-auto-java

Usage: Windows only.

Enables the "auto-java" feature. The auto-java feature enables automatic start of the Java app agent for all new JVMs. It directs the Universal Agent to inspect the command line arguments of each new process and add the "-javaagent:..." argument to all new Java processes. This option only modifies the command lines for processes that are recognized as JVMs. This option does not directly impact the background Universal Agent daemon.

This feature is not supported on Windows 2008 SP2 32-bit and 64-bit.

--disable-auto-java

Disables the auto-java feature that was previously enabled with the --enable-auto-java argument. This command line option does not directly impact the background Universal Agent daemon.

--enable-ldpreload

Usage: Linux only. Exactly like --enable-auto-java, but on Linux.

--disable-ldpreload

Usage: Linux only. Exactly like --disable-auto-java. Disables the auto-java feature that was previously enabled with the --enable-ldpreload argument. This command line option does not directly impact the background Universal Agent daemon.
--uninstall

Uninstalls the Universal Agent. The Universal Agent Daemon process is stopped and, if the Universal Agent is defined as a service, the service definition is deleted. On Windows, the Universal Agent definition is removed from the registry. Requires reboot of the machine.
Universal Agent Rulebooks

The Universal Agent uses rulebooks to manage the deployment and maintenance of runtime agents. This topic introduces rulebooks and strategies for applying them.

Working with Rulebooks

The Universal Agent operates according to rules that you define in rulebooks. A rulebook is a JSON-formatted configuration file that can direct the Universal Agent to install, stop, or start runtime agents.

The rulebook contains general properties, subject to condition evaluation logic, for the Universal Agent itself, along with rules for individual types of runtime agents.

So for instance, when you add a runtime agent rule to the rulebook, the Universal Agent to which the rule applies retrieves the runtime agent from an agent repository and installs the agent in the `monitor` directory in the Universal Agent home. The `monitor` directory contains the base install directories for each runtime agent.

**Operation Mode**

The Universal Agent may run in one of two modes: controller mode or local mode. The mode it uses depends on the rulebooks that it finds in the rulebook directory, `<universal_agent_home>/rulebook`.

The following describes the rulebooks that apply in each mode:

- **Local mode**: If a rulebook file named `local.json` exists in the directory, the Universal Agent operates in local mode. It does not attempt to contact the Controller and `controller-book.json` is ignored if it exists. You can put the Universal Agent into local mode by using the Universal Agent CLI or by creating `local.json` manually.
- **Controller mode**: Otherwise, the Universal Agent operates in Controller mode. When in Controller mode, the Universal Agent tries to get the operative rulebook from the Controller. If successful, the Universal Agent writes the contents of the Controller rulebook to a file named `controller-book.json` in its own rulebook directory. If the `controller-book.json` file exists in the rulebook directory, from that read or a previous read, then it is used as the rulebook. Otherwise, it operates using the default `rulebook`.

For information on creating and managing controller rulebooks, see Universal Agent REST APIs.

**Polling Interval**

The Universal Agent reads the rulebook at regular intervals and applies changes in the rulebook as they occur, reporting the event to the Controller. By default, the Universal Agent checks the rulebook every 300 seconds (5 minutes).

For testing and initial investigatory work, you may want to reduce this interval to induce more frequent polling. Use the `interval` property found in `universalagent.yaml` to change the polling frequency.

**Rulebook Structure**

The basic parts of a rulebook are shown below in tabular format. The header section tab contains settings that apply globally to all runtime agents controlled by this rulebook. In this example, the values identify the Universal Agent version and the Controller connection settings. Global properties enable you to avoid repeating the property in each rule.

The rules section tab contains rules that define runtime agents, in this case, two Java Agents and one Machine Agent. The header section and each rule can contain a config object. Any value specified in the rule overrides the global value.
Header Section

The key-value pairs in this config object apply globally to the agents controlled by the rulebook.

```json
{ "name": "default",  
  "comments": "Controller Connection section",  
  "config": {  
    "version": "4.5.0.0",  
    "controller_host": "192.168.99.100",  
    "controller_port": "8080",  
    "account_name": "customer1",  
    "account_access_key": "c61f32-nnnn"
  }
}
```

Rules Section

The key-value pairs in these rule-level config objects apply only to the agent defined by the rule.
"rules": [{
  "name": "Machine Agent Rule",
  "comments": "Install and start the machine agent",
  "monitor": "machine",
  "condition": "True",
  "config": {
    "state": "started"
  }
},
{
  "name": "Java Agent Install Rule 1",
  "comments": "Inject the Java Agent as JVMs are started",
  "monitor": "java",
  "config": {
    "state": "installed",
    "application_name": "MyApp",
    "tier_name": "MyTier",
    "node_name": "MyNode"
  }
},
{
  "name": "Java Agent Install Rule 2",
  "comments": "Attach the Java Agent to running JVMs",
  "monitor": "java",
  "config": {
    "state": "attached",
    "application_name": "MyApp2",
    "tier_name": "MyTier2",
    "node_name": "MyNode2"
  }
},
{
  "name": "Universal Agent rule",
  "comments": "Universal Agent Default Rule",
  "monitor": "universal",
  "config": {
    "version": "4.5.0.0",
    "state": "started"
  },
  "condition": "True"
}]}
Rulebook Example

The following is a complete rulebook with both header and rules in JSON format:

```json
{
    "name": "default",
    "comments": "Controller Connection section",
    "config": {
        "version": "4.5.0.0",
        "controller_host": "192.168.99.100",
        "controller_port": "8080",
        "account_name": "customer1",
        "account_access_key": "c61f32-nnnn"
    },
    "rules": [
        {
            "name": "Machine Agent Rule",
            "comments": "Install and start the machine agent",
            "monitor": "machine",
            "condition": "True",
            "config": {
                "state": "started"
            }
        },
        {
            "name": "Java Agent Install Rule 1",
            "comments": "Inject the Java Agent as JVMs are started",
            "monitor": "java",
            "config": {
                "state": "installed",
                "application_name": "MyApp",
                "tier_name": "MyTier",
                "node_name": "MyNode"
            }
        },
        {
            "name": "Java Agent Install Rule 2",
            "comments": "Attach the Java Agent to running JVMs",
            "monitor": "java",
            "config": {
                "state": "attached",
                "application_name": "MyApp2",
                "tier_name": "MyTier2",
                "node_name": "MyNode2"
            }
        }
    ]
}
```
{
    "name": "Universal Agent rule",
    "comments": "Universal Agent Default Rule",
    "monitor": "universal",
    "config": {
        "version": "4.5.0.0",
        "state": "started"
    },
    "condition": "True"
Rulebook Property Reference

**Header Rulebook Properties**

- **name**: Name for this rulebook.
- **comments**: An optional description for this rulebook.
- **config**: This object contains global properties that apply to all runtime agents in the rulebook. The `config` object notably contains connection properties for the Controller. See **Controller Connection Properties** for more information.
- **rules**: Rules entries contain the specific rules for controlling one or more agents. There are common properties in the agent rules and properties that are specific to the agent type. See **Runtime Agent Rule Properties** for more information.

The following sections provide more information on the `config` object and `rules` sections of the rulebook.

**Controller Connection Properties**

The rulebook example above shows properties that runtime agents use to connect to the Controller and how the agent instance is identified in the Controller UI. These properties correspond to values typically configurable for the runtime agents, particularly for the Java app agent configuration file, `controller-info.xml`.

Using the Universal Agent, you can set any of the usual properties found in the runtime agent configuration file. In the Universal Agent rulebook, the properties have the same names that are in `controller-info.xml`, except that a hyphen in the `controller-info.xml` name is replaced by an underscore in the rulebook version of the property name.

Specifically, the rulebook supports the following properties:

- `account_access_key`
- `account_name`
- `agent_runtime_dir`
- `application_name`
- `controller_host`
- `controller_port`
- `controller_ssl_enabled`
- `credential_store_filename`
- `credential_store_password`
- `enable_orchestration`
- `force_agent_registration`
- `machine_path`
- `node_name`
- `tier_name`
- `use_encrypted_credentials`
- `use_simple_hostname`

For more information on usage, see inline comments in the `controller-info.xml` file and the topic: **Administer App Server Agents**.

When you set the connection values in the rulebook, the Universal Agent updates the `controller-info.xml` on disk for the runtime agent. The following values are taken from `universalagent.yaml` if they are not specified in the rulebook:

- `controller_host`
- `controller_port`
- `controller_ssl_enabled`
- `account_name`
- `account_access_key`

Specifying the values in the rulebook, however, enables you to configure the Universal Agent and its configured runtime agents to talk to different Controllers, if necessary.

**Runtime Agent Rule Properties**

The rules section in the Universal Agent rulebook contains rules governing the presence and status of a runtime agent on the Universal Agent host. The default rulebook, `default.json`, is installed with the following predefined rule:
"rules": [ 
{
  "name": "Universal Agent rule",
  "comments": "Universal Agent rule",
  "monitor": "universal",
  "config": { 
    "version": "4.5.0.0",
    "state": "started"
  },
  "condition": "True"
}
]

The properties for each rule are:

- **name**: Name for this rule.
- **comments**: An optional description for this rule.
- **monitor**: The agent type. Valid values are:
  - `java` for the Java Agent
  - `machine` for the Machine Agent
  - `universal` for the Universal Agent
  - `dotnet` for the .NET Agent
  - `analytics` for the Analytics Agent
  - `network` for the Network Agent
- **condition**: A statement that must be true to be applied by a particular Universal Agent. Typically a "condition" tests an environment variable or system property on the host on which the Universal Agent runs.
- **config**: A JSON Object containing agent-level configuration properties such as operating state, node name, and so on. These can be specified per runtime agent in the rule (as illustrated by the state and application_name properties in the sample) or globally (as illustrated by the version property in the header section). Values specified in the rule override the global value. The config properties include:
  - **version**: The AppDynamics version of the app agent to use
  - **state**: The action to be applied to the agent, such as `installed`, which installs the agent or `started`, which both installs and starts it. For agent specific state information, see Java Agent Rules and Standalone Machine Agent Rules.
  - **application_name**, **tier_name**, and **node_name**: Applicable to Java Agents, these properties are equivalent to `application-name`, `tier-name`, and `node-name` in `controller-info.xml` for the traditional agent configuration. They specify the business application, tier, and node by which the current monitored process is identified in the Controller UI. For additional connection related properties, see Controller Connection Properties. Other runtime agents may have agent-specific properties that can be defined in the rule-level config object.

### Rulebook Strategies

#### Using Groups

Universal Agent groups are a way to manage multiple Universal Agents as a logical group. By default, Universal Agents are part of the default group and run the default rulebook, `default-controller`. As additional Universal Agents start and register with the controller, they join the default group.

You can create groups and add Universal Agents to them using the Universal Agent REST API.

A Universal Agent can be part of multiple groups. When the Universal Agent is added to the groups and those groups have different rulebooks, the Controller sends multiple rulebooks to the Universal Agent and the rulebooks are logically merged into a single rulebook.

The resulting merged rulebook is written to `controller-book.json`. You should not modify this rulebook directly.
**Conditional Rules**

You can use conditions to specify criteria for applying rules. In the simplest (and default) case, the value can simply be set to "true" to enable the rule as shown in the example:

```
"condition": "True"
```

You can also create test conditions that enable or disable the rule based on the environment in the Universal Agent host.

Condition property syntax in Universal Agent rulebooks is similar to conditions in Python—the condition should evaluate to a Boolean expression with identification data keys as the operands (which should match the regular expression "[a-zA-Z0-9_]+".

To install only on Linux machines and given the “platform_system” environment variable, you could specify the following condition:

```
"condition": "platform_system == '\''Linux'\''"
```

Notice that single quotes in the value are escaped using '\'.

In the following example, the condition checks for Linux and a version of the Universal Agent higher than 4.5.1:

```
"condition": "platform_system == '\''Linux'\'' and universalagent_version > '\''4.5.1'\''"
```

Operators you can use are:

- Logical operators: and, or, not
- Comparison operators: ==, !=, <, <=, >, >=
- Special operators:
  - Regular expression compare, where regular expression comes after the operator
  - ~ (AWK-like regular expression match)
  - !~ (AWK-like regular expression inverse match)
- Membership test: in

Instead of using dynamically evaluated conditions, you can put the static values of true or false in the condition property. When used in this way, the condition property gives you a convenient way to enable or disable individual rules.

**Configuration Templates**

Configuration templates enable you to define a set of default configuration values that apply across a set of rulebooks. See Rulebook Configuration Templates for details.

**Uninstalling a Runtime Agent**

To uninstall a runtime agent, remove the rule for it in the rulebook and save the file.

The Universal Agent first stops the runtime agent, if it is running, and then uninstalls the runtime agent from the application or machine and removes the directory for it.
Universal Agent Rules

On this page:
- Define Universal Agent Rules
- Example Universal Agent Rule
- Using Multiple Universal Agent Rules

In addition to maintaining runtime agents, the Universal Agent can manage itself through the rulebook, as described in this topic.

Define Universal Agent Rules

In a Universal Agent rulebook, a monitor value of universal identifies a Universal Agent rule.

```
...
"monitor": "universal",
...
```

The state property has the following values for the Universal Agent:

- **installed**: The Universal Agent binaries for the specified version should be downloaded and installed.
- **started**: The Universal Agent binaries for the specified version should be download and installed, and that this version should be running

When you install the Universal Agent, you get a default rulebook that includes a rule for the Universal Agent.

To upgrade the Universal Agent, you simply load the new version of the agent in the repository and increment the version number in the rule. When the Universal Agent detects that the rulebook specifies a different version than the version currently running, it automatically restarts itself with the new version.

Example Universal Agent Rule

The default rulebook includes a default rule for the Universal Agent, as follows:

```
{
  "name": "Universal Agent rule",
  "comments": "Universal Agent rule",
  "monitor": "universal",
  "config": {
    "version": "4.5.0.0",
    "state": "started"
  },
  "condition": "True"
}
```

Using Multiple Universal Agent Rules

You can have multiple Universal Agent rules in a rulebook. Each should specify a different name and a different version. Only one rule
at a time can specify a state of `started`.

If multiple rules specify a state of `started`, only the first one is recognized, and the other `started` Universal Agent rules are ignored. As with other rules in the rulebook, Universal Agent rules are ignored if the condition expression yields a value of `False`. 
Standalone Machine Agent Rules

The AppDynamics Universal Agent uses a rulebook to determine which versions of the runtime agents should be installed and deployed. You manage the deployment, versioning, and status of Standalone Machine Agents on a monitored machine by adding Machine Agent rules to the Universal Agent rulebook. This topic describes the syntax and usage for Machine Agent rules.

Define Standalone Machine Agent Rules

A `monitor` value of `machine` identifies a Machine Agent rule.

The valid values for the `state` property for a Machine Agent rule are:

- `installed`
- `started`

The started state both installs the agent (if not installed yet) and starts it. The `sim_enabled` property enables Server Visibility for the machine agent. When a value is not specified in the rule, the value defaults to `true`. See Server Visibility for details on licensing and functionality.

The other properties shown in the example, such as `name` and `comments`, are common across the agent types. For those rule properties, see Universal Agent Rulebooks.

Example Standalone Machine Agent Rule

The rule containing `"name": "Machine monitor"` specifies the configuration for a Machine Agent.
[{
    "name": "default-controller",
    "comments": "An example rule book with a rule to start a machine monitor",
    "config": {},
    "rules": [
      {
        "config": {
          "state": "started",
          "version": "4.5.0.0",
          "sim_enabled": ["true"|"false"]
        },
        "monitor": "machine",
        "comments": "This is a Machine rule",
        "condition": "True",
        "name": "Machine monitor"
      },
      {
        "name": "Universal Agent rule",
        "comments": "Universal Agent rule",
        "monitor": "universal",
        "config": {
          "version": "4.5.0.0",
          "state": "started"
        },
        "condition": "True"
      }
    ]
}]
Java Agent Rules

The AppDynamics Universal Agent uses a rulebook to determine which versions of the runtime agents should be installed and deployed. You manage the deployment, versioning, and status of Java Agents on a monitored machine by adding Java Agent rules to the Universal Agent rulebook. This topic describes the syntax and usage for those Java Agent rules.

**Define Java Agent Rules**

The `monitor` value of `java` in the Universal Agent rule identifies it as a Java Agent rule.

A Java Agent rule contains properties common to all Universal Agent rules, such as the name, description and monitor type of the agent. It also includes properties that are unique to Java Agents.

One example of a Java Agent rule is the following:

```json
{
  "name": "Java Rule 1",
  "comments": "Java agent rule",
  "monitor": "java",
  "config": {
    "version": "4.5.0.0",
    "state": "started",
    "application_name": "My Application",
    "tier_name": "Commerce Tier",
    "node_name_prefix": "$network_hostname.Commerce",
    "deploy_cmd": ".*-Dnode_type=commerce.*",
    "do_not_deploy_cmd": ".*-Dproduction=true.*"
  }
}
```

The following ‘Deployment Options’ section provides more information on using the state property. See the [Java Agent Rule Syntax Reference](#) for information about the other fields in the example.

**Deployment Options**

Using the `state` property, you can direct the Universal Agent to deploy the Java Agent using one of the following approaches:

- **Remote attach**: Universal Agent installs the Java Agent and if `auto-java` is enabled, automatically deploys the Java Agent into the JVM at startup time. If the target JVM is already running, then the Java Agent tries to remotely attach itself to the JVM without forcing a restart, regardless of whether auto-java is enabled.
- **At JVM startup**: Universal Agent installs the Java Agent and if `auto-java` is enabled, deploys the Java Agent into the JVM at JVM startup. If the JVM is already running, the agent does not try to remotely attach to it, so a restart is required to deploy the Java Agent.
- **Install manually**: Universal Agent installs the Java Agent, but does not deploy it to a JVM. In this approach, you deploy to a
JVM by manually adding the Java Agent to the JVM startup routine.

Attach the Java Agent into Running JVMs

Remote attach enables the Java Agent to load itself into a running JVM, without a JVM restart or modification of the JVM startup commands.

When using this approach, take note of the following points:

- The same requirements apply to using remote attach with the Universal Agent as apply when using it directly with the Java Agent. These requirements include the use of Oracle HotSpot JVM, for example. See Install the Java Agent for more information on these requirements.
- The user under which the Universal Agent runs, must adhere to the following conditions:
  - On Windows, the Universal Agent must run under the Local System account.
  - On Linux, the Universal Agent must run as one of the following users:
    - root
    - the same user that runs the JVMs
    - a user authorized to use sudo. See more information at Permissions for Running the Universal Agent.

Remote attach can affect the performance of the application during the attachment initialization process. Be sure to use this feature only after careful testing in a staging environment.

To attach the Java Agent into running JVMs:

1. Ensure that tools.jar is in the JAVA_HOME/lib/ext directory on the monitored hosts.
2. In the Java Agent rule of the Universal Agent rulebook, set the value of the state property to attached.
3. If the Universal Agent is not running as root or as the same user that runs the JVM, authorize the user for sudo privileges, as described in Permissions for Running the Universal Agent.
4. Specify conditions for attaching the Java Agent to your JVMs by using the deploy_cmd and deploy_env_vars properties in the Java Agent rule. For example, you can set conditions on installation based on startup commands that match Tomcat startup commands.
5. Save the rulebook. The Universal Agent applies the rule change the next time it reads the active configuration.

Install the Java Agent Automatically at JVM Startup

When you set the state to started, the Universal Agent examines each new process started on the host. If the process is a new JVM, the Universal Agent injects the javaagent argument into the startup arguments for the process. With this approach, you do not need to modify the Java startup script and it is supported on all JVM types.

Since this affects the JVM startup process, JVMs that are already running will not be affected. Also, the Java Agent version must be 4.2.3 or later.

AppDynamics Java Agent versions 4.2.6.x and 4.2.7.x include an issue that prevents the Universal Agent from being able to install those versions directly. The issue was addressed in Java Agent version 4.2.8.

If you are using version 4.2.6.x or 4.2.7.x, you can work around this incompatibility by having the Universal Agent first install Java Agent 4.2.8, or a later version. You do not need to deploy the later version to your application; simply create a rule for it with a state value of installed, while specifying attached or started as the state for the 4.2.6.x or 4.2.7.x agent rules.

If you are testing this feature on Linux, keep in mind, this feature relies on the Linux LD_PRELOAD facility, along with environment variables set in the JVM startup environment. Command shells that are open before you have enabled auto-java may not have the environment variables properly set.

On Windows, the auto-java feature is analogous to the LD_PRELOAD feature on Linux. When the auto-java feature is enabled, the Universal Agent gains visibility to the command line arguments for any new JVM process being started and adds two new arguments to the command line as follows:

- -javaagent:<FQPN_javaagent.jar>, where <FQPN_javaagent.jar> is the fully-qualified path name of the javaagent.jar file, as installed by the Universal Agent. The addition of this command line argument causes the Java agent to be loaded into the new JVM
- -Dappdynamics.ua.rule.dir=<FQPN_UA_rule_dir>, where <FQPN_UA_rule_dir> is the fully-qualified path name of the directory containing the Universal Agent rulebook.

If the Java command line arguments already contain the -javaagent argument, referring to javaagent.jar, or javaagent.jar cannot be located, then the Universal Agent does not modify the Java command line.
To insert the Java Agent into the JVM at JVM start up

1. Specify `started` as the value of the `state` property for the matching Java agent rule in the rulebook:
   a. Set the `state` property for the matching Java agent rule in the rulebook to started.
   b. When using the "started" (or auto-java) mode, it is likely that you will want to specify conditions for attaching the Java Agent to JVMs.
      For example, you can set conditions on installation based on startup commands that match Tomcat startup commands. Use the `deploy_cmd` and `deploy_env_vars` configuration attributes in the Java Agent rule for this type of control.

2. Enable the auto-java feature by executing the following command from a command prompt:

   ```
   ua --enable-auto-java
   ```

3. When you save the rulebook, the Universal Agent applies the rule change the next time it reads the active configuration.

See Universal Agent CLI for more details on the auto-java command line arguments.

**Install the Java Agent Manually**

Instead of relying on the Universal Agent to install the Java Agent into JVMs, you can modify the startup scripts for the JVM to inject the Java Agent. This is the traditional method for installing Java Agents, as described in Install the Java Agent. However, by using the Universal Agent, you can rely on it to manage the download and maintenance of the agent on the host.

The Universal Agent CLI tool lets you inspect the arguments for a given Java process. You can use this to determine any other command line options that need to be added in addition to the `javaagent` option. Use the CLI command `ua --show-java-arguments <pid>` argument—where `<pid>` is the process ID of the target application JVM—to see what command line options are required.

To install the Java Agent but inject into JVMs manually:

1. Set the value of the `state` property for the matching Java agent rule in the rulebook to `installed`.
2. Add the Java Agent JAR file to the startup routine of the JVM, as described for your Java framework type in Install the Java Agent. The agent JAR should be the Java Agent located in the `<ua_home>/monitor/java` directory.

**Java Agent Rule Matching**

A rulebook can have multiple Java Agent rules. When applying rules, the Universal Agent applies the first rule that matches the JVM, ignoring other possibly matching rules.

The `condition` property also affects rule selection. If the evaluation of `condition` yields a value of `False`, the rule is not applied for a JVM.

Because a single rule can match multiple JVMs, you should use the auto-node naming feature to ensure that each matching JVM is given a unique node name. To do this, do not include the `node_name` property in the rule. Instead, either provide a `node_name_prefix` property to provide a node name prefix, or omit the `node_name_prefix` property to enable the Controller to generate a node name using the default prefix. In either case, a unique node name is generated for the JVM.

**Java Agent Rule Syntax Reference**

The syntax of the Java Agent rule is illustrated in the following rulebook entry:
{  
    "name": "<name of rule>",
    "comments": "<comments>",
    "monitor": "java",
    "config": {
        "version": "<Java agent version>",
        "state": "installed | started | attached",
        "controller_host": "<name of host running controller>",
        "controller_port": "<controller primary port>",
        "account_name": "<controller account name>",
        "account_access_key": "<controller account access key>",
        "application_name": "<business application>",
        "tier_name": "<tier name>",
        "node_name": "<node name>",
        "node_name_prefix": "<node name prefix for auto-naming>",
        "deploy_cmd": "<regular expression matching Java command line>",
        "deploy_env_vars": "<regular expression matching JVM environmental variables>",
        "do_not_deploy_cmd": "<regular expression matching Java command line to be excluded>",
        "crash_age_threshold_days": "<prevents installation to JVMs with recent crash logs>",
        "runtime_directory": "<identifies the runtime directory for the Java agent...>",
        "additional_props": "<additional properties to be passed to the JVM...>",
        "kind": "<indicates whether or not IBM version of Java agent is to be installed...>",
        "condition": "<rule status>" "<boolean expression indicating status of this rule>"
    }
}

The name, comments, monitor, and condition properties are common for all Universal Agent monitor rules, and have no special meaning for the Java Agent. See Universal Agent Rulebooks for information on these values.

The config object contains properties specific to the Java Agent. The following table describes the config properties.

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td>The version of the Java Agent to run.</td>
<td>&quot;version&quot;: &quot;4.5.0.0&quot;</td>
</tr>
<tr>
<td>state</td>
<td>Indicates how the Java Agent deploys itself into the target JVM. Also related to the auto-java feature. Valid values are installed, started and attached. For more information, see Deployment Options.</td>
<td>&quot;state&quot;: &quot;started&quot;</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
<td>Value</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>controller_host</td>
<td>The name of the controller host where the Java Agent should connect.</td>
<td>&quot;controller_host&quot;: &quot;localhost&quot;</td>
</tr>
<tr>
<td></td>
<td>Optional. Can be inherited from the controller_host property defined in the header section of the rulebook or from the Controller identified in the universalagent.yaml file.</td>
<td></td>
</tr>
<tr>
<td>controller_port</td>
<td>The Controller port number where the Java Agent should connect.</td>
<td>&quot;controller_port&quot;: &quot;8080&quot;</td>
</tr>
<tr>
<td></td>
<td>Optional. Can be inherited from the controller_host property defined in the header section of the rulebook or from the Controller identified in the universalagent.yaml file.</td>
<td></td>
</tr>
<tr>
<td>account_name</td>
<td>The account name that is passed to the Controller when the Java Agent attempts to connect.</td>
<td>&quot;account_name&quot;: &quot;customer1&quot;</td>
</tr>
<tr>
<td></td>
<td>Optional. Can be inherited from the controller_host property defined in the header section of the rulebook or from the Controller identified in the universalagent.yaml file.</td>
<td></td>
</tr>
<tr>
<td>account_access_key</td>
<td>The account access key that is passed to the Controller when the Java Agent attempts to connect.</td>
<td>&quot;account_access_key&quot;: &quot;SJ5b2m7d1$354&quot;</td>
</tr>
<tr>
<td></td>
<td>Optional. Can be inherited from the controller_host property defined in the header section of the rulebook or from the Controller identified in the universalagent.yaml file.</td>
<td></td>
</tr>
<tr>
<td>application_name</td>
<td>The application name to be provided by the Java Agent when it connects to the controller.</td>
<td>&quot;application_name&quot;: &quot;ACME Book Store Application&quot;</td>
</tr>
<tr>
<td></td>
<td>Optional. Can be inherited from the header section of the rulebook defined in a global config object or from the Controller identified in the universalagent.yaml file.</td>
<td></td>
</tr>
<tr>
<td>tier_name</td>
<td>The tier name to be provided by the Java Agent when it connects to the controller.</td>
<td>&quot;tier_name&quot;: &quot;ECommerce Server&quot;</td>
</tr>
<tr>
<td></td>
<td>Optional. Can be inherited from the header section of the rulebook defined in a global config object or from the Controller identified in the universalagent.yaml file.</td>
<td></td>
</tr>
<tr>
<td>node_name</td>
<td>The node name to be provided by the Java Agent when it connects to the controller.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Optional. If this property is missing and the node_name_prefix property is provided, then the node_name_prefix is used to name the node of the JVM.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>If both the node_name and the node_name_prefix properties are missing, then the auto-node name feature is used to name the node. The controller provides a unique name for the node, using the tier name as a prefix for the node name.</td>
<td>&quot;node_name&quot;: &quot;Node_8000&quot;</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
<td>Example</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>node_name_prefix</td>
<td>Indicates that the Java Agent should use the &quot;auto node name&quot; feature to define a unique name dynamically for the JVM. The value of the &quot;node_name_prefix&quot; property specifies a prefix for the generated node name; the controller supplies a suffix to make the node name unique. If the &quot;node_name&quot; property is specified, then the &quot;node_name_prefix&quot; property is ignored, and the &quot;node_name&quot; property explicitly names the node. In the example Java Agent rule, &quot;$network_hostname.Commerce&quot; serves as a prefix. Notice that &quot;$network_hostname&quot; refers to a Universal Agent environment value, which the Universal Agent binds to a value at runtime. (Environment variable references always begin with '$'). If both the &quot;node_name&quot; or the &quot;node_name_prefix&quot; properties are missing, then the &quot;auto node name&quot; feature is used to name the node. The controller provides a unique name for the node, using the tier name as a prefix for the node name.</td>
<td></td>
</tr>
<tr>
<td>unique_host_id</td>
<td>The value of the &quot;appdynamics.agent.uniqueHostId&quot; property. Optional. Can be inherited from the header section of the rulebook (defined in a global &quot;config&quot; object) or from the Controller identified in the universalagent.yaml file.</td>
<td></td>
</tr>
<tr>
<td>deploy_cmd</td>
<td>A regular expression that identifies the JVMs that the rule applies to. This regular expression is matched against the command line arguments that are used to start the JVM. If the expression matches the command line, then this rule is considered a candidate for deploying the Java Agent within the JVM. If the deploy_env_vars property is also provided, then the value of that property must also match the set of environmental variables associated with the JVM. If the do_not_deploy_cmd property is also provided, then JVMs whose command line arguments match the deploy_cmd are not considered candidates for deployment if the command line also matches the value of the do_not_deploy_cmd property. In the example Java Agent rule, the deploy_cmd regular expression is looking for a Java property &quot;node_type=commerce&quot;. If the command line for a starting JVM matches the regular expression, it is considered a candidate for the rule. (However, because the do_not_deploy_cmd property is also specified in this example, the rule is not applied to a JVM if it's command line matches the value of deploy_cmd but also matches the value of do_not_deploy_cmd.</td>
<td></td>
</tr>
</tbody>
</table>

"node_name_prefix": "Commerce"

"unique_host_id": "cart-machine"

"deploy_cmd": ".*/TIER1TOMCAT/.*"
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>deploy_env_vars</td>
<td>A regular expression that identifies the JVMs that the rule applies to. The set of environmental variables associated with the JVM is formed into a string, consisting of &quot;key=value&quot; pairs, delimited by blanks, and sorted alphabetically based on the key name. If the resulting string matches the regular expression defined by the &quot;deploy_env_vars&quot; property, then this rule is considered a candidate for deploying the Java Agent within the JVM. If this property is not provided, then the contents of the environmental variables are not used to select JVM(s) for deployment. If the &quot;deploy_cmd&quot; property is also provided, then the value of that property must also match the command line used to start the JVM. If the &quot;do_not_deploy_cmd&quot; property is also provided, then JVMs whose command line arguments match the &quot;deploy_cmd&quot; property are not considered candidates for deployment if the command line also matches the value of the &quot;do_not_deploy_cmd&quot; property.</td>
<td>&quot;deploy_env_vars&quot;: &quot;.<em>SHOULD_APPLY=true.</em>&quot;</td>
</tr>
<tr>
<td>do_not_deploy_cmd</td>
<td>A regular expression that is used to exclude JVMs that the rule should not apply to. This regular expression is matched against the command line arguments being used to start the JVM. If the expression matches the command line, then this rule is explicitly excluded from being considered a candidate for deploying the Java Agent within the JVM. In the example Java Agent rule, the regular expression will match those Java command lines that contain &quot;-Dproduction=true&quot;. JVMs whose command lines match this expression are excluded from deployment for this rule.</td>
<td>&quot;do_not_deploy_cmd&quot;: &quot;.<em>-Ddo_not_apply=true.</em>&quot;</td>
</tr>
<tr>
<td>crash_age_threshold_days</td>
<td>Indicates that the Universal Agent should not remotely attach itself to a JVM where there are one or more recent Java crash log files. The value of this property represents the number of days since the crash log file was created in order to be considered 'recent'. For example, if a value of 10 is specified for this property, then only crash log files created in the last 10 days prevent the Universal Agent from attempting remote attach. Crash logs older than 10 days do not prevent the attempted attach. If this property is not specified, then the Universal Agent does not look for recent Java crash log files, and does not use their presence to prevent remote attach attempts.</td>
<td></td>
</tr>
<tr>
<td>additional_props</td>
<td>A comma-separated list of key=value pairs providing additional property definitions to be passed to the JVM. Passed to the JVM if the value of the state keyword is specified as either &quot;started&quot; or &quot;attached&quot;.</td>
<td>&quot;additional_props&quot;: &quot;appdynamic.agent.uniqueHostId=host&quot;</td>
</tr>
<tr>
<td>kind</td>
<td>Use this property when the IBM version of the Java agent is being deployed. The value should be &quot;ibm&quot;. If the property is missing, then the non-IBM version of the agent is deployed.</td>
<td>&quot;kind&quot;: &quot;ibm&quot;</td>
</tr>
</tbody>
</table>
Dynamic Configuration Values

Configuration settings can be set to static or dynamic values. Dynamic values may be bound to environment variables or system properties from the target application JVM itself. This lets you name nodes, tiers, or applications based on JVM-specific variables. This is particularly useful for elastic environments, in which node or tier identity cannot be known in advance.

To reference dynamic elements in rules, use the following format:

- For Java environmental variables, prefix the variable name with `$_javaenv_`
- For system properties, use the syntax `${property-name}`, where `property-name` is the name of the Java system property.

When determining whether or not a Java rule applies to a particular JVM, the Universal Agent extracts the values of each environment variable, and perform symbolic substitution into the config values in the rule for each `$javaenv_` reference. The Universal Agent does not resolve Java system properties within the configuration rule. Instead, the references are resolved by the Java Agent itself when it starts running within the JVM.

Example

The following shows a rule with sample dynamic values:

```json
{
    "name": "Java rule",
    "comments": "...comments...",
    "monitor": "java",
    "config": {
        "version": "4.5.0.0",
        "state": "attached",
        "application_name": "$_javaenv_APP_NAME_",
        "tier_name": "$_tierName_",
        "node_name": "$_javaenv_APP_NAME_-$javaenv_NODE_NAME_",
        "deploy_cmd": "...regular expression matching Java command line..."
    },
    "condition": "...boolean expression indicating status of rule..."
}
```

In this example, several of the properties have values referencing either JVM environment variables or system properties:

- `application_name`: `$_javaenv_APP_NAME_` indicates that the value of the `application_name` config property is the value of the `APP_NAME` environmental variable within a JVM for which the rule applies.
- `tier_name`: `$_tierName_` indicates that the value of the `tier_name` config property is the value of the `tierName` system property within a JVM for which the rule applies.
- `node_name`: `$_javaenv_NODE_NAME_` indicates that the value of the `node_name` config property is the value of the `APP_NAME` environmental variable within a JVM for which the rule applies, concatenated with a `-'`, followed by the value of the `NODE_NAME` environmental variable within the JVM.

Suppose a JVM is started with the following commands:
export APP_NAME=MyApplication
export NODE_NAME=MyNode

java -DtierName=MyTier  -jar app.jar

If matched to the sample rule, the result would the following configuration bindings for the JVM:

- "application_name": "MyApplication"
- "tier_name": "MyTier"
- "node_name": "MyApplication-MyNode"

Notes on Dynamic Configuration

- References to Java system properties are not resolved by the Universal Agent when evaluating rule conditions, since the property values from the JVM are not available to the Universal Agent.
- For environment variable substitution, if a Java Agent is introduced into a JVM at startup time (as opposed to remote attach), the javaagent.jar file must be version 4.3 or later. Note that the Java Agent must be installed by the Universal Agent, but it does not need to be deployed—installing the Java Agent causes the supporting javaagent.jar file to be loaded into each JVM.
You can install, maintain, and run .NET Agents with the AppDynamics Universal Agent. The Universal Agent uses a rulebook to determine which versions of the .NET Agent should be installed and deployed. You need to define and configure the .NET Agent entry in the Universal Agent rulebook.

**Define .NET Agent Rules**

A monitor value of `dotnet` identifies a .NET Agent rule.

The valid values for the `state` property for a .NET Agent rule are:

- installed
- started

**.NET Agent Rule Syntax**

```
{
    "condition": "platform_family == 'windows'",
    "comments": "...comments...",
    "name": "...name of rule...",
    "monitor": "dotnet",
    "config": {
        "version": "...NET agent version...",
        "state": "installed | started",
        "controller_host": "...name of host running controller...",
        "controller_port": "...port number controller is listening on...",
        "account_name": "...controller account name...",
        "account_access_key": "controller account access key...",
        "controller_ssl_enabled": "true",
        "controller_tls12_enabled": "false",
        "controller_secure_enabled": "false",
        "config_xml": [
            "list of repositories where a paired config.xml can be found"
        ]
    }
}
```

The following properties are defined for the Universal Agent and common to all agent types. The values have no special meaning for the .NET agent:

- **name**: A string for quickly identifying the rule.
- **comments:** A long string describing the rule’s purpose.
- **monitor:** Name of the targeting monitoring agent, for .NET Agent, must be dotnet.
- **condition:** The conditions for applying the rule. This consists of a Boolean expression where the operands are the values collected by the Environment Modules.

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td>The version of the .NET Agent to run.</td>
<td>&quot;version&quot;: &quot;4.5.0.0&quot;</td>
</tr>
<tr>
<td>state</td>
<td>Indicates how the .NET Agent should deploy on the target machine.</td>
<td>&quot;state&quot;: &quot;started&quot;</td>
</tr>
<tr>
<td></td>
<td>- &quot;installed&quot;: downloads the specified version of the .NET Agent</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- &quot;started&quot;: a version of the .NET Agent is installed and is active</td>
<td></td>
</tr>
<tr>
<td>controller_host</td>
<td>Identifies the name of the Controller host where the agent should connect.</td>
<td>&quot;controller_host&quot;: &quot;localhost&quot;</td>
</tr>
<tr>
<td></td>
<td>This property is optional in the rule; it can be inherited</td>
<td></td>
</tr>
<tr>
<td></td>
<td>from the controller_host property defined in the header section of the</td>
<td></td>
</tr>
<tr>
<td></td>
<td>rulebook or from the Controller identified in the universalagent.yaml file.</td>
<td></td>
</tr>
<tr>
<td>controller_port</td>
<td>Identifies the port number of the Controller where the agent should connect.</td>
<td>&quot;controller_port&quot;: &quot;8080&quot;</td>
</tr>
<tr>
<td></td>
<td>This property is optional in the rule; it can be inherited</td>
<td></td>
</tr>
<tr>
<td></td>
<td>from the controller_host property defined in the header section of the</td>
<td></td>
</tr>
<tr>
<td></td>
<td>rulebook or from the Controller identified in the universalagent.yaml file.</td>
<td></td>
</tr>
<tr>
<td>account_name</td>
<td>Identifies the account name that is passed to the Controller when</td>
<td>&quot;account_name&quot;: &quot;customer1&quot;</td>
</tr>
<tr>
<td></td>
<td>the agent attempts to connect. This property is optional in the rule; it</td>
<td></td>
</tr>
<tr>
<td></td>
<td>can be inherited from the controller_host property defined in the header</td>
<td></td>
</tr>
<tr>
<td></td>
<td>section of the rulebook or from the Controller identified in the</td>
<td></td>
</tr>
<tr>
<td></td>
<td>universalagent.yaml file.</td>
<td></td>
</tr>
<tr>
<td>account_access_key</td>
<td>Identifies the account access key that is passed to the Controller when</td>
<td>&quot;account_access_key&quot;: &quot;SJ5b2m7d1$354&quot;</td>
</tr>
<tr>
<td></td>
<td>the agent attempts to connect. This property is optional in the rule; it</td>
<td></td>
</tr>
<tr>
<td></td>
<td>can be inherited from the controller_host property defined in the header</td>
<td></td>
</tr>
<tr>
<td></td>
<td>section of the rulebook or from the Controller identified in the</td>
<td></td>
</tr>
<tr>
<td></td>
<td>universalagent.yaml file.</td>
<td></td>
</tr>
<tr>
<td>controller_ssl_enabled</td>
<td>SSL is enabled</td>
<td>&quot;controller_ssl_enabled&quot;: &quot;true&quot;</td>
</tr>
<tr>
<td>controller_enable_tls12</td>
<td>tls12 is enabled</td>
<td>&quot;controller_enable_tls12&quot;: &quot;false&quot;</td>
</tr>
<tr>
<td>controller_secure</td>
<td>Secure is enabled</td>
<td>&quot;controller_secure&quot;: &quot;false&quot;</td>
</tr>
<tr>
<td>config_xml</td>
<td>Specifies, in JSON format, a list of repository URLs where config.xml,</td>
<td>&quot;config_xml&quot;: [repository,</td>
</tr>
<tr>
<td></td>
<td>related scripts, and binaries are located.</td>
<td>... ] where repository=server</td>
</tr>
<tr>
<td></td>
<td>See Configuring the .NET Agent.</td>
<td>URL, such as <a href="http://server/path">http://server/path</a></td>
</tr>
<tr>
<td></td>
<td>file://local/path</td>
<td>or file://local/path</td>
</tr>
</tbody>
</table>

**Example .NET Agent Rule**

In this .NET Agent rule example, the Controller and account settings are not shown because they are inherited from the global Universal Agent config or from the universalagent.yaml file.
Configuring the .NET Agent

The Universal Agent installs the .NET Agent and deploys the configuration file (config.xml) specified in the rulebook. The .NET Agent uses the config.xml file to determine which processes to instrument upon restart of the process. It is up to you to build the config.xml file and specify its location using the config_xml property in the .NET rule. A sample setup configuration file can be viewed in Unattended Installation for .NET.

For each .NET Agent version, you can specify a corresponding value for the config.xml property. This config file is retrieved from the repository specified by the config_xml property. The location can be local to the machine in a shared director, or can be external.

```json
{
    "condition": "platform_family == 'windows'",
    "comments": "This is an .NET Agent rule",
    "name": " .NET Agent rule for Windows",
    "monitor": "dotnet",
    "config": {
        "state": "started",
        "version": "4.5.0.0",
        "config_xml": [
            "file:///C:/Tools/Repo"
        ]
    }
}
```
Analytics Agent Rules

On this page:
- Define Analytics Agent Rules
- Example for Machine Agent Mode
- Example for Standalone Mode

Related pages:

The AppDynamics Universal Agent uses a rulebook to determine which versions of the runtime agents should be installed and deployed. You can manage the deployment, versioning, and status of Analytics Agents collecting Transaction Analytics on a monitored machine by adding Analytics Agent rules to the Universal Agent rulebook. This topic describes the syntax and usage for the Analytics Agent rules.

The supported functionality includes:

- Running the Analytics Agent as either an extension to the Standalone Machine Agent or in a standalone JVM
- Configuring the Analytics Agent property files, to run as either an extension to the Standalone Machine Agent or in a standalone JVM as indicated with the `mode` property
- Upgrading the standalone Analytics Agents to new versions
- Downgrading standalone Analytics Agents to prior versions
- Using the Analytics Agent to collect Transaction Analytics

You typically run the Analytics Agent in standalone mode if you are collecting only analytics data from the target machine. If you also need to collect hardware metrics or need to run other Machine Agent extensions, you would use the machine agent mode.

When deploying the Analytics Agent on Linux using the Universal Agent, Standalone mode is not supported, you must use machine-agent mode.

If you have an existing Machine Agent running outside of Universal Agent control and you want to install the Analytics Agent, you should either manage both with the Universal Agent, or manage each separately (not using the Universal Agent).

Define Analytics Agent Rules

A `monitor` value of `analytics` identifies an Analytics Agent rule.

The valid values for the `state` property for an Analytics Agent rule are:

- `installed`
- `started`

Analytics Agent Rules Syntax

The syntax for an Analytics Agent rulebook entry is as follows:
```
{
    "name": "...name of rule...",
    "comments": "...comments...",
    "monitor": "analytics",
    "config": {
        "mode": "machine-agent" | "standalone",
        "version": "...Analytics agent version...",
        "state": "installed" | "started",
        "controller_host": "...name of host running controller...",
        "controller_port": "...port number controller is listening on...",
        "account_name": "...controller account name...",
        "global_account_name": "...global account name...",
        "account_access_key": "controller account access key...",
        "deploy_in": [ "<machine-agent-rule-name>" , ... ],
        "props": [ "<prop-name-1>": "<prop-value-1>" , "<prop-name-2>": "<prop-value-2>" , ... , "<prop-name-n>": "<prop-value-n>" ],
        "vmoptions": [ "<vmoption-1>" , "<vmoption-2>" , ... , "<vmoption-n>" ],
        "force_recycle": "true" | "false",
        "condition": "...boolean expression indicating status of rule..."
    }
}
```

The name, comments, monitor, and condition properties are common to all agent rules and have no special meaning for the Analytics Agent:

- **name**: A short string to identify the rule.
- **comments**: A long string describing the rule’s purpose.
- **monitor**: Name of the target runtime agent (for Analytics Agent, must be “analytics”).
- **condition**: Conditions for applying this rule. A condition consists of a Boolean expression where the operands are the values collected by the environment modules.

The properties within the **config** object have specific meanings for the Analytics Agent. They identify the following:

- Analytics Agent execution mode
- Property values for the Universal Agent to use to override the default values in the analytics property files
- Network location of the Controller that the agent connects with
- Whether or not changes to the Analytics Agent configuration should cause the hosting Machine Agent to be recycled.

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>mode</td>
<td>Indicates whether the analytics agent is to run as a machine agent (machine-agent) or as a standalone JVM (standalone)</td>
<td>&quot;mode&quot;: &quot;machine-agent&quot;</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
<td>Value</td>
</tr>
<tr>
<td>--------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>version</td>
<td>Identifies the version of the analytics agent to run. This is applicable only if the mode is &quot;standalone&quot;. When running as a machine agent extension, the analytics agent uses the same version as the machine agent.</td>
<td>&quot;version&quot;: &quot;4.5.0.0&quot;</td>
</tr>
<tr>
<td>state</td>
<td>Indicates the state of the analytics agent:</td>
<td>&quot;state&quot;: &quot;started&quot;</td>
</tr>
<tr>
<td>controller_host</td>
<td>Identifies the name of the host that the controller the agent should connect to is running on. This property is optional in the rule; it can be inherited from the controller_host property defined in the header section of the rulebook or from the Controller identified in the universalagent.yaml file.</td>
<td>&quot;controller_host&quot;: &quot;localhost&quot;</td>
</tr>
<tr>
<td>controller_port</td>
<td>Identifies the port number that the controller the agent should connect to is listening on. This property is optional in the rule; it can be inherited from the controller_host property defined in the header section of the rulebook or from the Controller identified in the universalagent.yaml file.</td>
<td>&quot;controller_port&quot;: &quot;8080&quot;</td>
</tr>
<tr>
<td>account_name</td>
<td>Identifies the account name that is passed to the controller when the agent attempt to connect. This property is optional in the rule; it can be inherited from the controller_host property defined in the header section of the rulebook or from the Controller identified in the universalagent.yaml file.</td>
<td>&quot;account_name&quot;: &quot;customer1&quot;</td>
</tr>
<tr>
<td>global_account_name</td>
<td>Identifies the global account name that is used to populate the http.event.accountName property in the analytics-agent.properties file. This property is optional in the rule; it can be inherited from the controller_host property defined in the header section of the rulebook or from the Controller identified in the universalagent.yaml file.</td>
<td>&quot;global_account_name&quot;: &quot;customer1_5c5826ff-0cc6-4745-8f25-af757d3a510d&quot;</td>
</tr>
<tr>
<td>account_access_key</td>
<td>Identifies the account access key that is passed to the Controller when the agent attempts to connect. This property is optional in the rule. The account access key can be inherited from the controller_host property defined in the header section of the rulebook or from the Controller identified in the universalagent.yaml file.</td>
<td>&quot;account_access_key&quot;: &quot;SJ5b2m7d1$354&quot;</td>
</tr>
<tr>
<td>events_services_host</td>
<td>Identifies the network name or address of the Events Service host. This value is used to populate the <code>http.event.endpoint</code> property in the <code>analytics-agent.properties</code> file. If this property is missing, then the value of the &quot;controller_host&quot; property is used.</td>
<td>&quot;event_services_host&quot;: &quot;1.2.3.4&quot;</td>
</tr>
<tr>
<td>events_services_port</td>
<td>Identifies the port that the Events Service is listening on. This value is used to populate the <code>http.event.endpoint</code> property in the <code>analytics-agent.properties</code> file. If this value is missing, then the default value 9080 is used.</td>
<td>&quot;event_services_port&quot;: &quot;444&quot;</td>
</tr>
<tr>
<td>deploy_in</td>
<td>Applies only when the mode is <code>machine-agent</code>. This property identifies the Machine Agent where the Analytics Agent runs as an extension. The value is a list of one or more strings, each of which contains the name of a Machine Agent rule within the Universal Agent rulebook. This argument is required if &quot;mode&quot;: &quot;machine-agent&quot; is specified.</td>
<td>&quot;deploy_in&quot;: [ &quot;machine-agent-1&quot; ]</td>
</tr>
<tr>
<td>props</td>
<td>Contains one or more property definitions. Each property definition is added (or modified, if the property already exists) in the <code>conf/analytics-agent.properties</code> file for the analytics agent.</td>
<td>&quot;props&quot;: { &quot;http.event.error.retryAttempts&quot;: &quot;500&quot; }</td>
</tr>
<tr>
<td>vmoptions</td>
<td>Contains one or more JVM options definitions. Each JVM option is added to the <code>conf/analytics-agent.vmoptions</code> file.</td>
<td>&quot;vmoptions&quot;: [ &quot;-verbose:class&quot; ]</td>
</tr>
<tr>
<td>force_recycle</td>
<td>If true is specified, then the Universal Agent restarts any Machine Agents hosting this Analytics Agent if any of the configuration arguments change. The default is false. Applicable only if &quot;mode&quot;: &quot;machine-agent&quot; is specified.</td>
<td>&quot;force_recycle&quot;: &quot;true&quot;</td>
</tr>
</tbody>
</table>

**Example for Machine Agent Mode**

This example specifies that the Analytics Agent should run as a Machine Agent extension. In this example:

- **mode**: `machine-agent` indicates machine-agent mode
- **state**: "started" indicates that the Analytics Agent should be started within the Machine Agent.
- **deploy_in**: Identifies the Machine Agents where the Analytics Agent should run.
- **props**: Defines a property that will be added to, or modified within, the `analytics-agent.properties` file of the Analytics Agent.
- **force_recycle**: A value of true indicates that the hosting Machine Agent should be recycled if the any of the config properties of the Analytic Agent change.

The following example illustrates these properties.
Example for Standalone Mode

This example specifies that the Analytics Agent should run in a standalone JVM. In this example:

- **mode**: standalone indicates standalone mode.
- **state**: started indicates that the Analytics Agent should be started
- **props**: Defines a property that will be added to, or modified within, the analytics-agent.properties file of the Analytics Agent.

The following is an example of the Analytics Agent rules:

```
{
    "mode": "standalone",
    "version": "4.5.0.0",
    "state": "started",
    "props": { "http.event.error.retryAttempts": "500" }
}
```

Special Considerations for Standalone mode

When you specify "mode": "standalone" for your Analytics Agent rule, there are some special considerations:

- Although a rulebook can contain multiple entries defining a standalone Analytics Agent, only one Analytics Agent runs at a time. When multiple rules are present that specify "state": "started", the agent defined by the first rule is started. Analytics Agents identified by subsequent rules are not started. If other agents are currently running, they are stopped.
- If any configuration properties change for an Analytics Agent that is currently running, that agent is restarted.
Network Agent Rules

Define Network Agent Rules

The Universal Agent can download, install, configure, and deploy the Network Agent. The Universal Agent uses a rulebook to determine which version of the Network Agent should be installed and deployed. You need to define and configure the Network Agent rule in the Universal Agent rulebook.

A monitor value of network identifies a Network Agent rule.

Valid values for the Network Agent rule's state property are:

- "installed": Downloads and installs the Network Agent.
- "started": Starts the Network Agent. (Also installs the Network Agent if it's not already installed.)
- "stopped": Stops the Network Agent.

Add to the Source Repository

When you place the Network Agent in the source repository, the installation distribution must have the following format:

```
network-<version>-<word_size>-<os>.zip
```

- version: Agent version. For example, 4.5.0
- word_size: 32bit or 64bit
- os: linux

Network Agent Syntax

Network Agent rulebook is similar to the following.
{  
  "name": "network",
  "comments": "Example Network Agent rulebook",
  "config": {
  
  },
  "rules": [
    
    "name": "Network Agent rule",
    "comments": "Install and start the Network Agent",
    "monitor": "network",
    "condition": "True",
    "config": {
      "version": "4.5.0.0",
      "state": "started"
    }
  
    
  ]
}
Dynamic Configuration Values

On this page:
- Types of Environment Variables
- How to Set Environment Variables
- Example Rulebook with Dynamic Configuration Values

In rulebooks, you can set configurations with either static or dynamic values. Static values are set with literals, such as a literal string, whereas, dynamic values are based on environment variables. For most rulebook configurations, using static values is sufficient. For an elastic environment where the identity of nodes or tiers cannot be known in advance, however, you may want to use dynamic configuration values to name nodes, tiers, or applications.

The following sections will discuss the types of environment variables, how they are referenced in rulebooks, which agents have access to them, and how they are set. The last section provides a rulebook example using dynamic values for the configurations for rules for both the Java and Machine Agents.

Types of Environment Variables

The Universal Agent can access the three types of environment variables shown in the table below to set dynamic configuration values.

<table>
<thead>
<tr>
<th>Type</th>
<th>Syntax</th>
<th>Example</th>
<th>Agents with Access to Environment Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java environment variables</td>
<td>$javaenv_&lt;variable_name&gt;</td>
<td>$javaenv_APP_NAME</td>
<td>Java Agent</td>
</tr>
<tr>
<td>Java system properties</td>
<td>&quot;${property-name}&quot;, where property-name is the name of the Java system property.</td>
<td>${tierName}</td>
<td>Java Agent</td>
</tr>
<tr>
<td>system environment variables</td>
<td>$env_&lt;variable_name&gt;</td>
<td>$env_TIER_NAME</td>
<td>Java Agent, Machine Agent</td>
</tr>
</tbody>
</table>

How to Set Environment Variables

The following sections describe the process for defining, setting, and applying different types of environment variables.

Java Environment Variables

To set Java environment variables, you define export environment variables and then start the JVM with the environment variables as command-line arguments. The JVM will then set the Java environment variables so that they are accessible to the Universal Agent. If the Java rule applies to a particular JVM, the Universal Agent then extracts the values of each environment variable and performs symbolic substitution into the config values in the rule for each $javaenv_ reference.

The flow for setting and applying Java environment variables is illustrated in the diagram below:
Java System Properties

As with Java environmental variables, you can also set Java system properties by passing command-line arguments when you start the JVM. In addition, you can programmatically set Java system properties. If the Java rule applies to a particular JVM, the Universal Agent extracts the values of each environment variable and performs symbolic substitution into the config values in the rule for each \$\{system-property\} reference.

The flow for setting and applying Java system properties is illustrated in the diagram below:

For environment variable substitution, if a Java Agent is introduced into a JVM at startup (as opposed to being remote attached), the javaagent.jar file must be version 4.3 or later. The 4.3 Java Agent must be installed by the Universal Agent, but it does not need to be deployed—simply installing the 4.3 Java Agent causes the supporting javaagent.jar file to be loaded into each JVM.
As mentioned above, you can also programmatically set a Java system property through the `System` class as shown below.

```java
Properties props = System.getProperties();
props.setProperty("tierName", "MyTier");
```

### System Environment Variables

The method of setting system environment variables and starting the Unified Agent depends on the version of the machine OS. The general process is to set system environment variables in a configuration file and then use the system-specific daemon or the CLI to start the Unified Agent. If a rule applies to either a particular JVM or system, the Universal Agent extracts the values of each system environment variable and performs symbolic substitution into the config values in the rule for each `$env_` reference.

The flow for setting and applying system environment variables is illustrated in the diagram below:
Example Rulebook with Dynamic Configuration Values

The following example rulebook has rules containing dynamic configuration values for both Java and Machine Agents. For more information about the dynamic configuration values, see the explanation following the example.

Note: The env and javaenv variables can be used in condition fields. In condition fields you should leave off the $ sign from the env and javaenv variables. Review the java rule in the rulebook that follows for an example.
{
    "name": "AD-Capital",
    "comments": "Agents for AD-Capital",
    "config": {},
    "rules": [
        {
            "name": "Java rule",
            "comments": "...comments...",
            "monitor": "java",
            "config": {
                "version": "4.5.0.0",
                "state": "attached",
                "application_name": "$env_APP_NAME",
                "tier_name": "${tierName}",
                "node_name": "$javaenv_APP_NAME-$javaver_NODE_NAME",
            },
            "condition": "env_DEPLOYMENT=="PROD""
        },
        {
            "name": "machine-4.5.0.0",
            "monitor": "machine",
            "comments": "Update Machine Agent to 4.5.0.0",
            "config": {
                "state": "started",
                "version": "4.5.0.0",
                "application_name": "$env_APP_NAME",
                "tier_name": "$env_TIER_NAME",
                "node_name": "$env_NODE_NAME"
            },
            "condition": "platform_system == '"Linux"' and universalagent_version > '"4.5.0"'"
        },
        {
            "name": "Universal Agent rule",
            "monitor": "universal",
            "comments": "Update Universal Agent to 4.5.0.0",
            "config": {
                "state": "started",
                "version": "4.5.0.0"
            },
            "condition": "True"
        }
    ]
}
In the previous rulebook example, the rule for the Java Agent contains the following attributes having values referencing JVM environment variables, Java system properties, and system environment variables:

- "application_name": "$env_APP_NAME": Indicates that the value of the application_name config attribute is the value of the APP_NAME environment variable set by the system (daemon/CLI) for which the rule applies.
- "tier_name": "${tierName}": Indicates that the value of the tier_name config attribute is the value of the tierName system property within a JVM for which the rule applies.
- "node_name": "$javaenv_APP_NAME-$javaver_NODE_NAME": Indicates that the value of the node_name config attribute is the value of the APP_NAME environment variable within a JVM for which the rule applies, concatenated with a '-' followed by the value of the NODE_NAME environment variable within the JVM.
- "env_DEPLOYMENT": This variable is used in the condition field. This checks whether there is an environment variable DEPLOYMENT set with a value equal to PROD. Note that although the env and javaenv variables can be used in condition fields, the format is different. You should not specify the $ sign.

The rule for the Machine Agent contains the following attributes having values referencing system environment variables:

- "application_name": "$env_APP_NAME": Indicates that the value of the application_name config attribute is the value of the APP_NAME environment variable set by the system (daemon/CLI) when the Universal Agent is started, for which the rule applies.
- "tier_name": "$env_TIER_NAME": Indicates that the value of the tier_name config attribute is the value of the TIER_NAME environment variable set by the system (daemon/CLI) when the Universal Agent is started, for which the rule applies.
- "node_name": "$env_NODE_NAME": Indicates that the value of the node_name config attribute is the value of the NODE_NAME environment variable set by the system (daemon/CLI) when the Universal Agent is started, for which the rule applies.
Rulebook Configuration Templates

Configuration templates enable you to define a set of default configuration values that apply across a set of rulebooks. The configuration templates are stored on the AppDynamics Controller and used to update the rulebooks on machines where the Universal Agent manages the runtime agents. The use of configuration templates is optional, however, using them can reduce the size of your rulebooks and simplify rulebook management across a set of machines.

**Background**

The rulebook for each Universal Agent instance uses a JSON object called `config` to define a set of key-value pairs containing configuration values that are specific for the agent that is being monitored. These `config` objects can be large and in many cases may contain redundant fields across numerous runtime agents that are being managed by a single Universal Agent instance.

Configuration templates implement a JSON object that defines a basic, default configuration for your runtime agents. The template object is a set of key-value pairs that can be referenced in your rulebooks at both the header level (applying globally to all runtime agents in the rulebook) and in agent-specific rules.

The configuration template is applied to the rulebook first, and properties in the `config` object in the rulebooks can override the default values if needed for specific use cases.

Users with the Configure Universal Agent permission can create and update configuration templates using the REST APIs.

- Configuration templates can be used only when running the Universal Agent in **controller mode**.
- The rulebooks reference applicable configuration templates using the `configEntity` property.
- Individual rules in your rulebook can override values defined in the templates using the rule-level `config` object.
- When the Universal Agent processes a rule that contains a `configEntity` property, a request is sent to the Controller for the content of the specified template. The template values are then pushed to the Universal Agent, where they are applied to the rulebook.
- The Universal Agent reports the version ID of each rulebook and template that was sent in the prior cycle back to its Controller. If necessary, the Controller then sends more recent versions of the rulebooks and templates back to the Universal Agent.
Workflows for Using Configuration Templates

On this page:

- Specify Global Configuration Values in a Template
- Placeholder for Rule Configurations in Rulebooks
- Reusing Configuration Templates

Related pages:

- Rulebook Configuration Templates
- Universal Agent Rulebooks
- Universal Agent REST APIs

Knowledge and understanding of the Universal Agent rulebook structure is a prerequisite to understanding and using configuration templates. Review the topic: Universal Agent Rulebooks before using configuration templates.

Specify Global Configuration Values in a Template

When you have common configuration values across different rulebooks, instead of repeating all the fields in a header config section in each separate rulebook, you can create a configuration template that contains the common values and then add a reference to the template in the relevant rulebooks.

This scenario uses the Configuration Template to specify configuration values that apply across multiple rulebooks. The configuration values in the template will be incorporated into the referring rulebook.

1. Create the configuration template using the Configuration Template PUT API. The following tabs show a sample command and the response format.
### Create Configuration Template

```shell
curl -i -X PUT -su '
'<controller-user>@<controller-account>:<controller-password>' -H "Content-type: application/json" -H "Accept:application/json"
"http://<controller-host:controller-port>/controller/universalagent/v1/user/configurations/controllerConfig" --data
'{
  "name": "controllerConfig",
  "config": {
    "controller_host": "<controller-host>",
    "controller_port": "<controller-port>",
    "account_name": "<controller-account>",
    "account_access_key": "<account-access-key>"
  },
  "model": "NameValue"
}
```

**Response Payload**

Created template

```json
{
  "name": "controllerConfig",
  "config": {
    "controller_host": "host.controller.com",
    "controller_port": "8080",
    "account_name": "customer1",
    "account_access_key": "Acces$$Key"
  },
  "model": "NameValue",
  "referredRuleBooks": []
}
```
1. HTTP Response Code
200 OK

Response Fields

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Value</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the configuration template</td>
<td></td>
</tr>
<tr>
<td>model</td>
<td>Model of the template</td>
<td>NameValue (only supported model)</td>
</tr>
<tr>
<td>config</td>
<td>Configuration</td>
<td>Name Value pairs</td>
</tr>
<tr>
<td>referredRuleBooks</td>
<td>List of rulebooks referring this template</td>
<td>Read only</td>
</tr>
</tbody>
</table>

2. Create a rulebook that refers to the configuration template using the rulebook PUT API and the `configEntity` keyword. The following tabs show a sample command and the response format.

Command
curl -i -X PUT -su
"http://<<controller-host>>:<controller-port>/controller/universalagent/v1/user/rulebooks/ByName/JavaRuleBook" --data
'{
   "name": "JavaRuleBook",
   "comments": "An example rule book with a rule to start a java agent monitor",
   "configEntity": "controllerConfig",
   "rules": [
   {
      "name": "java-monitor",
      "comments": "Rule to start java agent",
      "monitor": "java",
      "config":
      {
         "version": "4.5.0.0",
         "state": "started",
         "application_name": "Test-Application",
         "tier_name": "tier1",
         "node_name_prefix": "node",
         "deploy_cmd": ".*-Dcatalina.base=.*"
      },
      "condition": "True"
   }
   ]
}'

Response Payload
Created rulebook
```json
{
    "name": "JavaRuleBook",
    "comments": "An example rule book with a rule to start a java agent monitor",
    "configEntity": "controllerConfig",
    "config": {},
    "version": 1,
    "rules": [
    {
        "name": "java-monitor",
        "comments": "Rule to start java agent",
        "monitor": "java",
        "config": {
        "version": "4.5.0.0",
        "state": "started",
        "application_name": "Test-Application",
        "tier_name": "tier1",
        "node_name_prefix": "node",
        "deploy_cmd": ".*-Dcatalina.base=.*"
        },
        "condition": "True"
    }
    ]
}
```

HTTP Response Code
200 OK

Response Fields

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Value</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the rulebook</td>
<td>The recognizable name for the rulebook contents</td>
</tr>
<tr>
<td>comments</td>
<td>Any comments provided</td>
<td></td>
</tr>
<tr>
<td>configEntity</td>
<td>Configuration Template reference</td>
<td>If no template present, an empty string (&quot;&quot; ) will be returned.</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------------</td>
<td>-------------------------------------------------------------</td>
</tr>
<tr>
<td>config</td>
<td>Old style configuration</td>
<td>The returned value will be whatever name-value pair was provided when the rulebook was created. If no config field was provided, the returned value will be <code>{ }</code>.</td>
</tr>
<tr>
<td>version</td>
<td>A read-only number that changes every time the rulebook is modified</td>
<td>Changing the configuration template increments this version number too.</td>
</tr>
<tr>
<td>rules</td>
<td>List of rules for individual agents</td>
<td>If config values are provided in the rules, they override corresponding values from the configuration template.</td>
</tr>
</tbody>
</table>

This scenario is the equivalent of having the following config in the rulebook:

```
"config": {
    "controller_host": "host.controller.com",
    "controller_port": "8080",
    "account_name": "customer1",
    "account_access_key": "Access$$Key"
}
```

If both config and configEntity are provided in a rulebook, the resulting configuration is computed by over-riding configEntity fields with config fields.

For example, if the rulebook request shown in step 2 above, contains the config shown here in Config section and also specifies the "configEntity": "controllerConfig" property, the result is equivalent to what is shown in Config block:
**Placeholder for Rule Configurations in Rulebooks**

Configuration templates can also be used as placeholders for individual rule configurations. This means a "configEntity":<configuration_template_name> can be used to replace the "config" key-value pairs.

1. Create the configuration template. The tabs show a sample command and the response format.
Create Configuration Template

```
curl -i -X PUT -su
'<controller-user>@<controller-account>:<controller-password>' -H "Content-type: application/json" -H
"Accept:application/json" \
"http://<controller-host:controller-port>/controller/universalagent/v1/user/configurations/javamonitorconfiguration" \
--data '{
   "name": "javamonitorconfiguration",
   "config":
       {
           "version": "4.5.0.0",
           "state": "started",
           "application_name": "Test-Application",
           "tier_name": "tier1",
           "node_name_prefix": "node",
           "deploy_cmd": ".*-Dcatalina.base=.*"
       },
   "model": "NameValue"
}

Response Payload

Created template
```


```json
{
  "name": "javamonitorconfiguration",
  "config":
    {
      "version": "4.5.0.0",
      "state": "started",
      "application_name": "Test-Application",
      "tier_name": "tier1",
      "node_name_prefix": "node",
      "deploy_cmd": ".*-Dcatalina.base=.*"
    },
  "model": "NameValue"
  "referredRulebooks": []
}
```

HTTP Response Code
200 OK

Response Fields

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Value</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the configuration template.</td>
<td></td>
</tr>
<tr>
<td>model</td>
<td>Model of the template</td>
<td>NameValue (only supported)</td>
</tr>
<tr>
<td>config</td>
<td>Configuration</td>
<td>NameValue pairs</td>
</tr>
<tr>
<td>referredRuleBooks</td>
<td>List of RuleBooks referring this Template</td>
<td>Read only</td>
</tr>
</tbody>
</table>

2. Create or update a rulebook which refers to the template. This example updates a previously created rulebook. You can use the same procedure to create a new rulebook with `configEntity` reference. The tabs show a sample command and the response format.
Command

curl -i -X PUT -su
'\<controller-user>\<controller-account>:\<controller-password>\' -H "Content-type: application/json" -H "Accept:application/json" \\
"http://\<controller-host>:\<controller-port>/controller/universalagent/v1/user/rulebooks/byName/JavaRuleBook" --data \\
'\{
    "name": "JavaRuleBook",
    "comments": "An example rule book with a rule to start a java agent monitor",
    "configEntity": "controllerConfig",
    "rules": [
        {
            "name": "java-monitor",
            "comments": "Rule to start java agent",
            "monitor": "java",
            "configEntity": "javamonitorconfiguration",
            "condition": "True"
        }
    ]
'}

Response Payload

Created RuleBook
```json
{
    "name": "JavaRuleBook",
    "comments": "An example rule book with a rule to start a java agent monitor",
    "configEntity": "controllerConfig",
    "config": {},
    "version": 2,
    "rules": [
      {
        "name": "java-monitor",
        "comments": "Rule to start java agent",
        "monitor": "java",
        "configEntity": "javamonitorconfiguration"
      }
    ]
}
```
### configEntity Configuration Template reference

<table>
<thead>
<tr>
<th>config</th>
<th>Old style configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>This field will not be present if the <code>config</code> field is not provided in the request, else, it will be whatever name value pair provided. See Note that follows.</td>
<td></td>
</tr>
</tbody>
</table>

### Reusing Configuration Templates

The primary use of templates is for reuse across multiple rulebooks (or even rules). In this scenario, we create a rulebook with different controller settings but the same rule configuration as the Java rule in the previous example.

1. Create the Controller configuration template.

```bash
  "name": "OtherControllerConfig",
  "config": {
    "controller_host": "otherhost.controller.com",
    "controller_port": "8080",
    "account_name": "customer2",
    "account_access_key": "Acces$$Key2"
  },
  "model": "NameValue"
}'
```

### Response Format
OtherControllerConfig template created.

```json
{
   "name": "OtherControllerConfig",
   "config": {
      "controller_host": "otherhost.controller.com",
      "controller_port": "8080",
      "account_name": "customer2",
      "account_access_key": "Acces$$Key2"
   },
   "model": "NameValue"
}
```

2. Create a rulebook referring to OtherControllerConfig and sharing javamonitorconfiguration.

Command
curl -i -X PUT -su
'\<controller-user>\@\<controller-account>:\<controller-password>\' -H "Content-type: application/json" -H
"Accept:application/json" \
"http://\<controller-host>:\<controller-port>/controller/universalagent/v1/user/rulebooks/ByName/JavaRuleBook" --data
'{
  "name": "OtherJavaRuleBook",
  "comments": "An example rule book with a rule to start
a java agent monitor",
  "configEntity": "OtherControllerConfig",
  "rules": [
    {
      "name": "java-monitor",
      "comments": "Rule to start java agent",
      "monitor": "java",
      "configEntity": "javamonitorconfiguration"
      "condition": "True"
    }
  ]
}'

Response Payload

OtherJavaRulebook rulebook created.
3. View the shared configuration referredRuleBooks field by doing a GET controller/universalagent/v1/user/configurations/javamonitorconfiguration.

Command

curl -i -X GET -su
'<controller-user>:<controller-account>:<controller-password>'
-H "Content-type: application/json" -H
"Accept:application/json" \
"http://<controller-host:<controller-port>/controller/universalagent/v1/user/configurations/javamonitorconfiguration"

Response Payload

Note: referredRuleBooks shows both the referring rulebooks. You can see that javamonitorconfiguration
on is now shared between OtherJavaRuleBook and JavaRuleBook.

```json
{
  "name": "javamonitorconfiguration",
  "config": {
    "version": "4.5.0.0",
    "state": "started",
    "application_name": "Test-Application",
    "tier_name": "tier1",
    "node_name_prefix": "node",
    "deploy_cmd": ".*-Dcatalina.base=.*"
  },
  "model": "NameValue",
  "referredRuleBooks": ["OtherJavaRuleBook",
                        "JavaRuleBook"]
}
```
Universal Agent REST APIs

On this page:

- Permissions
- Using Environment Data
- API Rate and Total Count Limits
- Get Complete Information for Universal Agents
- Get Summary Information for Universal Agents
- Get Universal Agent By Name
- Delete Universal Agent by Name
- Create or Update a Group
- Add Universal Agent to a Group
- Delete Universal Agent From a Group
- Move Universal Agent to Multiple Groups
- Get a Group By Name
- Get Multiple Groups
- Delete a Group
- Create or Update a Rulebook
- Get Rulebooks
- Add a Rulebook to a Group
- Get a Rulebook Associated with a Group
- Delete a Rulebook By Name
- Delete a Rulebook From a Group
- Get All Events
- Get Event Details
- Get Event Details By Event ID
- Create a Configuration Template
- Update a Configuration Template
- Delete a Configuration Template
- View Configuration Templates

Related pages:

- Permissions for Running the Universal Agent
- Universal Agent Rulebooks
- Dynamic Configuration Values

The Universal Agent API is a Controller REST API that enables you to monitor and manage a Universal Agent deployment programmatically.

The Universal Agent API enables you to act on Universal Agent groups, individual Universal Agents, rulebooks, and Universal Agent events.

Permissions

The following RBAC permissions and roles authorize the use of the Universal Agent REST APIs.

- Permissions
  - Configure Universal Agent: Allows all PUT and DELETE APIs
  - View Universal Agent: Allows all GET APIs
- Roles
  - Universal Agent Administrator with both view and configure permissions
  - Universal Agent User with view permission

Universal Agent Administrator and Universal Agent User roles are added to all new and existing accounts. Also, Universal Agent Administrator and Universal Agent User roles are added to users who have Account Administrator role. Configure Universal Agent and View Universal Agent permissions are added to Account Administrator role of all accounts.

See Roles and Permissions for more information on AppDynamics permissions and roles.
Using Environment Data

Using the API calls described in Get Complete Information for Universal Agents, you can get information about the running environment for each Universal Agent and the runtime agents. You can then use the values to populate rulebook properties, such as tier and application names. See the topic Dynamic Configuration Values.

To read environment variable data from JVM environments, the Universal Agent needs to have sufficient permissions. See more information about how to provision the Universal Agent with sudo access if necessary in Permissions for Running the Universal Agent.

As an API response, the environment data is made up of the following name/value pairs:

- Information about the underlying platform:
  - platform_machine: Machine type. For example, i386
  - platform_processor: Processor name. For example, amdk6
  - platform_release: System/OS release. For example, 2.2.0 or NT
  - platform_system: System/OS name. For example, Linux or Windows
  - platform_version: System release version. For example, Darwin Kernel Version 14.4.0

- Network-related information:
  - hostname: Fully qualified domain name
  - ip_addresses: List of IP addresses for machines where the Universal Agent is running
  - mac_addresses: List of MAC addresses for machines where the Universal Agent is running

- Names and values of each environment variable. The keys are the name of each environment variable, prefixed with env_. For example, if the environment contains the variables HOME and SHELL, then this section may look like:
  - env_HOME: /Users/john.smith
  - env_SHELL: /bin/bash

- Time information:
  - time_utc: Current UTC date and time in ISO 8601 format

- Information about the Universal Agent
  - universalagent_version: Current version. For example, 4.5.0.0.
  - universalagent_build: Build number

- Information about the Java Agent:
  - java_targets: List of running Java processes on the machine. This is organized as a map of process IDs (PID) to process command line
  - java_installs: Absolute paths of the Java runtime agents installed by the Universal Agent. The path is organized as nested maps, using the app/tier/node name as the key. For example:
“java_installs”: {
    “MyApp”: {
        “MyTier”: {
            “MyNode1”: “/path/to/java/app/monitor/1”,
            “MyNode2”: “/path/to/java/app/monitor/2”
        },
    },
    “MyApp2”: {
        “MyTier2a”: {
            “Node”: “/path/to/java/app/monitor/3”
        },
        “MyTier2b”: {
            “Node”: “/path/to/java/app/monitor/4”
        },
        “null”: {
            “null”: “/path/to/java/app/monitor/that/only/had/app/specifield/in/controller/info/xml”
        }
    }
}

- Information about the Machine Agent
  - `machine_installs`: List of absolute paths to Machine Agents installed by the Universal Agent
  - `machine_starts`: List of absolute paths to the Machine Agents started by the Universal Agent

- Information about .NET Agent
  - `dotnet_frameworks`: List of the installed .NET frameworks
  - `dotnet_version`: .NET Agent version, if installed

- Information about the Network Agent
  - `none`

- Information about Analytics Agent
  - `none`

**API Rate and Total Count Limits**

The following rate limits apply by default:

- Maximum 5000 agent registrations per minute per account
- Maximum 5000 event registrations per minute per account
- Maximum 500 user requests per minute per account

The following total count limits apply by default:

- Maximum 5000 agents per account
- Maximum 500 rulebooks per account
- Maximum 500 groups per account

For on-premises Controllers, you can configure rate and total count limits from the Controller Administration Console.

**Get Complete Information for Universal Agents**
Returns information about running and previously running Universal Agents. When used with the `groups` query parameter, this API returns information only for the agents under the specified groups.

```
GET controller/universalagent/v1/user/agents?agents=<value1, value2>
GET /controller/universalagent/v1/user/agents?agents=<value1, value2>&groups=<value>
GET /controller/universalagent/v1/user/agents?groups=<value1, value2>
```

## Query Parameters

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Value</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>agents</td>
<td>A comma-delimited string of Universal Agent names. When present, returns the relevant matching subset of agents.</td>
<td>No</td>
</tr>
<tr>
<td>groups</td>
<td>A comma-delimited string of group names. When present, returns only agents under the specified groups.</td>
<td>No</td>
</tr>
</tbody>
</table>

## Example Command

```
curl -i -X GET -u '<userName>@<accountName>:<password>'
-H 'Content-type: application/json'
http://<controller-host>:8080/controller/universalagent/v1/user/agents
```

## Response Format

```
{
    "agent1": {
        "version": <string>,
        "ruleBookName": <string>,
        "environment": {
            "key1": <object>,
            ...
        }
    },
    ...
}
```

## Response Fields

- **agent1:** Name of the Universal Agent
- **version:** Property specifying the version of the Universal Agent
Get Summary Information for Universal Agents

View summary information about all running Universal Agents. The environment data is suppressed. When used with the `groups` query parameter, this API returns information only for the agents under the specified groups.

```
GET /controller/universalagent/v1/user/agents/summary?agents=<value>
GET /controller/universalagent/v1/user/agents/summary?agents=<value1,value2>&groups=<value>
GET /controller/universalagent/v1/user/agents/summary?groups=<value1,value2>
```

Query Parameters

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Value</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>agents</td>
<td>A comma-delimited string of Universal Agent names. When present, the relevant matching subset of agents is returned.</td>
<td>No</td>
</tr>
<tr>
<td>groups</td>
<td>A comma-delimited string of group names. When present, returns only agents under the specified groups.</td>
<td>No</td>
</tr>
</tbody>
</table>

Example Command

```
curl -i -X GET -u '<userName>@<accountName>:<password>'
-H 'Content-type: application/json'
http://<controller-host>:8080/controller/universalagent/v1/user/agents/summary
```

Response Format

```
{
    "agent1": {
        "version": <string>,
        "ruleBookName": <string>
    },
    ...
}
```
Response Fields

- `agent1`: Name of the Universal Agent
- `version`: Version of the Universal Agent
- `ruleBookName`: Name of the rulebook of the Universal Agent

Get Universal Agent By Name

View details for the Universal Agent with a specified name.

```
GET /controller/universalagent/v1/user/agents/byName/<name>
```

Path Parameters

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Value</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the Universal Agent. For example, <code>agent1</code>.</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Example Command

```
curl -i -X GET -u '<userName>@<accountName>:<password>'
-H 'Content-type: application/json'
http://<controller-host>:8080/controller/universalagent/v1/user/agents/byName/agent1
```

Response Format

```
{
    "version": <string>,
    "ruleBookName": <string>,
    "environment": {
        "key1": <object>,
        ...
    }
}
```

Response Fields

- `agent1`: Name of the Universal Agent
- `version`: Version of the Universal Agent
- `ruleBookName`: Name of the current rulebook of the Universal Agent
Delete Universal Agent by Name

Delete an agent by name.

```
DELETE /controller/universalagent/v1/user/agents/byName/<name>
```

Deleting an agent also deletes events associated with it.

Path Parameters

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Value</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the Universal Agent</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Example Command

```
curl -i -X DELETE -u '<userName>@<accountName>:<password>'
-H 'Content-type: application/json'
http://<controller-host>:8080/controller/universalagent/v1/user/agents/byName/agent1
```

Create or Update a Group

Universal Agent groups are a logical way of grouping different Universal Agents that are running on different machines. You can add different Universal Agents to a group and then add a common rulebook to that group. All Universal Agents in the group execute the same set of rules (that apply to their environment). For details on groups: see the section 'Rulebook Strategies' in Universal Agent Rulebooks

Create the group. If the group exists, it is updated.

```
PUT /controller/universalagent/v1/user/groups/byName/<name>
```

Path Parameters

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the group. In the example that follows, &quot;name&quot;: &quot;&quot; is overridden by the path parameter name</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Example Command
curl -i -X PUT -u '<userName>@<accountName>:<password>'
-H 'Content-type: application/json'
-H 'Accept:application/json'
http://<controller-host>:8080/controller/universalagent/v1/user/groups/byName/group1
-d '{"name":"","comments":"This is group1"}'

Response Format

{  
  "name": <string>,  
  "comments": <string>  
}

Add Universal Agent to a Group

Add the specified Universal Agent to the specified group.

PUT
/controller/universalagent/v1/user/agents/membership/<agentName>/groups/<groupName>

Path Parameters

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Value</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>agentName</td>
<td>Name of the Universal Agent</td>
<td>Yes</td>
</tr>
<tr>
<td>groupName</td>
<td>Group name. For example: group1</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Example Command
curl -i -X PUT -u '<userName>@<accountName>:<password>'
-H 'Content-type: application/json'
-H 'Accept:application/json'
http://<controller-host>:8080/controller/universalagent/v1/user/agents/membership/agent1/groups/group1

Response Format

```
{
   "agent1": {
      "version": <string>,
      "ruleBookName": <string>,
      "environment": {
         "key1": <object>,
         ...
      }
   },
   ...
}
```

Delete Universal Agent From a Group

Delete the specified Universal Agent from the specified group.

```
DELETE
/controller/universalagent/v1/user/agents/membership/<agentName>/groups/<groupName>
```

Path Parameters

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Value</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>agentName</td>
<td>Name of the Universal Agent. For example, agent1</td>
<td>Yes</td>
</tr>
<tr>
<td>groupName</td>
<td>Group name. For example, group1</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Example Command
curl -i -X DELETE -su '<userName>@<accountName>:<password> '
-H 'Content-type: application/json'
-H 'Accept:application/json'
http://<controller-host>:8080/controller/universalagent/v1/user/agents/membership/agent1/groups/group1

**Move Universal Agent to Multiple Groups**

Universal Agent is added to multiple groups and removed from previous groups.

PUT /controller/universalagent/v1/user/agents/groupsMembership/<agentName>

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Value</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>agentName</td>
<td>Name of the Universal Agent. For example, agent1</td>
<td>Yes</td>
</tr>
<tr>
<td>groupName</td>
<td>A list of group names formatted as JSON data. For example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-d '[[&quot;groupName&quot;:&quot;group1&quot;],[&quot;groupName&quot;:&quot;group2&quot;]]'</td>
<td></td>
</tr>
</tbody>
</table>

**Example Command**

curl -i -X PUT -su '<userName>@<accountName>:<password> '
-H 'Content-type: application/json'
-H 'Accept:application/json'
http://<controller-host>:8080/controller/universalagent/v1/user/agents/groupsMembership/agent1
-d '[["groupName":"group1"],["groupName":"group2"]]'

**Response Format**
Get a Group By Name

Return information for the named group.

```
GET /controller/universalagent/v1/user/groups/byName/<name>
```

Path Parameters

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the group. For example, group1</td>
<td>Yes</td>
</tr>
</tbody>
</table>

```
curl -i -X GET -u '<userName>@<accountName>:<password>'
-H 'Content-type: application/json'
http://<controller-host>:8080/controller/universalagent/v1/user/groups/byName/group1
```
Get Multiple Groups

Get all groups or specific groups, groups having specific agents or groups having specific rulebooks.

**GET**

```
/getter/universalagent/v1/user/groups?groups=group1,group2&agents=agent1,agent2&rulebooks=rulebook1
```

**Query Parameters**

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>groups</td>
<td>Use to get specific groups, such as /controller/universalagent/v1/user/groups?groups=group1,group2</td>
<td>No</td>
</tr>
<tr>
<td>agents</td>
<td>Use to get groups having specific agents.</td>
<td>No</td>
</tr>
<tr>
<td>rulebooks</td>
<td>Use to get groups specific rulebooks.</td>
<td>No</td>
</tr>
</tbody>
</table>

**Example Command**

```
curl -i -X GET -u '<userName>@<accountName>:<password>'
-H 'Content-type: application/json'
http://<controller-host>:8080/controller/universalagent/v1/user/groups
```

**Response Format**
Delete a Group

Deletes the group. Deletion of the default group named default is not allowed.

Format

DELETE /controller/universalagent/v1/user/groups/byName/<name>

Path Parameters

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the group, for example, group1.</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Example Command

```
curl -i -X DELETE -u '<userName>@<accountName>:<password>'
-H 'Content-type: application/json'
http://<controller-host>:8080/controller/universalagent/v1/user/groups/byName/group1
```

Create or Update a Rulebook

Create a rulebook. If a rulebook with "name" already exists, it is updated.

Format

PUT /controller/universalagent/v1/user/rulebooks/byName/<name>

Path Parameters

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Value</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the group. This path parameter overrides <em>name</em>: &quot;rulebook1&quot;</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Example Command
curl -i -X PUT -u '<userName>@<accountName>:<password>'
-H 'Content-type: application/json'
-H 'Accept:application/json'
http://<controller-host>:8080/controller/universalagent/v1/user/rulebooks
/byName/rulebook1
-d ' {
  "name": "rulebook1",
  "comments": "An example rulebook with a rule to start a machine agent",
  "config": {
    "controller_host": "localhost",
    "controller_port": "8080",
    "account_name": "",
    "account_access_key": ""
  },
  "rules": [
    {
      "name": "Example rule to start a 4.5.0.0 machine agent",
      "comments": "Enable by changing 'condition' to 'True'. Controller information is inherited from the 'config' section above. The machine agent will be installed in <universal_agent_dir>/monitor/machine/4.5/...",
      "monitor": "machine",
      "config": {
        "version": "4.5.0.0",
        "state": "started"
      },
      "condition": "True"
    }
  ]
}'

Response Format
Response Fields

- name: Name of the rulebook
- comments: Comments describing the rulebook
- config: Optional set of name-value configuration settings inherited by each rule or specified in a specific rule
- rules: Object with a list of rules in the rulebook

Get Rulebooks

View specific rulebooks by name or group or view a list of all rulebooks.

GET /controller/universalagent/v1/user/rulebooks
GET /controller/universalagent/v1/user/rulebooks/byName/<name>
GET /controller/universalagent/v1/user/rulebooks/current/<groupName>

Returns the name, comments, and the configuration rules.

Path Parameters

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Value</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the rulebook</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Example Command for a List of All Rulebooks

curl -i -X GET -u 'userName@accountName:password'
-H 'Content-type: application/json'
http://controller-host:8080/controller/universalagent/v1/user/rulebooks

Response Format

This version returns a list of rulebooks similar to the following:

```json
[
  {
    "name": <string>,
    "comments": <string>,
    "config": {
      "key1": <object>
      ...
    },
    "rules": [
      {
        "name": <string>,
        "comments": <string>,
        "monitor": <string>,
        "config": {
          "key1": <object>,
          ...
        },
        "condition": <string>
      },
      ...
    ]
  },
  ...
]
```

Example Command for Specific Rulebook by Name
curl -i -X GET -u '<userName>@<accountName>:<password>'
-H 'Content-type: application/json'
http://<controller-host>:8080/controller/universalagent/v1/user/rulebooks/byName/rulebook1

Example Command for Rulebooks in a Group

curl -i -X GET -u '<userName>@<accountName>:<password>'
-H 'Content-type: application/json'
http://<controller-host>:8080/controller/universalagent/v1/user/rulebooks/current/group1

Response Format

The versions using `name` and `groupName` return a single rulebook similar to the following:

```json
{
  "name": <string>,
  "comments": <string>,
  "config": {
    "key1": <object>
    ...
  },
  "rules": [
    {
      "name": <string>,
      "comments": <string>,
      "monitor": <string>,
      "config": {
        "key1": <object>,
        ...
      },
      "condition": <string>
    },
    ...
  ]
}
```
Response Fields

- name: Name of the rulebook
- comments: Comments describing the rulebook
- config: Optional set of name-value configuration settings inherited by each rule or specified in a specific rule
- rules: Object with a list of rules in the rulebook

Add a Rulebook to a Group

Indicate which rulebook should be used by the Universal Agents in the group indicated by groupName.

```
PUT /controller/universalagent/v1/user/rulebooks/current/<groupName>
```

Path Parameters

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Value</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>groupName</td>
<td>name of the group</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Example Command

```
curl -i -X PUT -u '"<userName>@<accountName>:<password>' -H 'Content-type: application/json' -H 'Accept:application/json' http://<controller-host>:8080/controller/universalagent/v1/user/rulebooks/current/group1 -d '"ruleBookName":"rulebook1"'
```

Response Format
Response Fields

- **name**: The name of the rulebook
- **comments**: Comments describing the rulebook
- **config**: Optional set of name-value configuration settings inherited by each rule or specified in a specific rule
- **rules**: Object with a list of rules for each agent in the rulebook

Get a Rulebook Associated with a Group

View the name of the current rulebook for the specified group of Universal Agents.

```
GET /universalagent/v1/user/rulebooks/current/<groupName>
```

Path Parameters

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Value</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>groupName</td>
<td>The name of the group.</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Example Command
curl -i -X GET -u '<userName>@<accountName>:<password>'
-H 'Content-type: application/json'
http://<controller-host>:8080/controller/universalagent/v1/user/rulebooks/current/group1

Response Format

```
{
    "name": <string>,
    "comments": <string>,
    "config": {
        "key1": <object>
        ...
    },
    "rules": [{
        "name": <string>,
        "comments": <string>,
        "monitor": <string>,
        "config": {
            "key1": <object>
            ...
        },
        "condition": <string>
    },
    ...
}
```

Response Fields

- **name**: Name of the rulebook
- **comments**: Comments describing the rulebook
- **config**: Optional set of name-value configuration settings inherited by each rule or specified in a specific rule
- **rules**: Section for a list of rules in the rulebook

Delete a Rulebook By Name

Remove the rulebook with the specified name.
Deleting a rulebook also removes it from any group association. Deletion of the default rulebook named "default-controller" is not allowed.

**Path Parameters**

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Value</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>name of the rulebook</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Example Command**

```
curl -i -X DELETE -u '<user_name>@<account_name>:<password>'  
-H 'Content-type: application/json'  
http://<controller_host>:8080/controller/universalagent/v1/user/rulebooks/byName/rulebook1
```

---

**Delete a Rulebook From a Group**

Remove a rulebook for Universal Agents by group.

```
DELETE /controller/universalagent/v1/user/rulebooks/current/<groupName>
```

Universal Agents without a rulebook execute local rulebooks (thus allowing manual rulebook creation). Deletion of the default rulebook named "default-controller" is not allowed.

**Path Parameters**

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Value</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>groupName</td>
<td>name of the group</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Example Command**
curl -i -X DELETE -u '<userName>@<accountName>:<password>'
-H 'Content-type: application/json'
http://<controller-host>:8080/controller/universalagent/v1/user/rulebooks/current/group1

Get All Events

View events that have occurred across all the Universal Agents. When the query parameters are present, the relevant matching subset of events is returned. By default, all events are shown up to the maximum limit of 600 events.

GET
/controller/universalagent/v1/user/events?agentIds=<value1,value2>&eventTypes=<value1,value2>&subTypes=<value>&timeRange=<value>

Query Parameters

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Value</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>agentIds</td>
<td>A comma-delimited strings of Universal Agent names. When present, events for the relevant matching subset of agents is returned.</td>
<td>No</td>
</tr>
<tr>
<td>eventTypes</td>
<td>A comma-delimited string of event types. When present, returns only relevant events. The valid event types are:</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>• UNIVERSAL_AGENT_STARTUP</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• UNIVERSAL_AGENT_SHUTDOWN</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• UNIVERSAL_AGENT_EXECUTE_RULE</td>
<td></td>
</tr>
<tr>
<td>subTypes</td>
<td>The success or error code. The valid subtypes are:</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>• Success</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Error</td>
<td></td>
</tr>
<tr>
<td>timeRange</td>
<td>Time range specifier string as used in other parts of the Controller. For example:</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>• last_1_hour.BEFORE_NOW.-1.1424287621109.60</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• last_5_minutes.BEFORE_NOW.-1.-1.5</td>
<td></td>
</tr>
</tbody>
</table>

Example Command

curl -i -X GET -u '<userName>@<accountName>:<password>'
-H 'Content-type: application/json'
http://<controller-host>:8080/controller/universalagent/v1/user/events
Response Format

```
[
  {
    "id":<long>,
    "version":<long>,
    "eventType":<string>,
    "eventTime":<Date>,
    "severity":<string>,
    "eventSummary":<string>,
    "affectedEntities": [
      {
        "id":<long>,
        "version":<long>,
        "entityType":<string>,
        "entityId":<long>,
        "prettyToString":<string>
      },...
    ],
    "correlationKeys":[],
    "properties":<string>,
    "markedAsRead":<boolean>,
    "markedAsResolved":<boolean>,
    "archived":<boolean>,
    "endTime":<long>,
    "guid":<string>,
    "subType":<string>,
    "triggeredEntity":{
      "id":<long>,
      "version":<long>,
      "entityType":<string>,
      "entityId":<long>,
      "prettyToString":<string>
    }
  },...
]
```

Response Fields:
- **id**: ID of the event
- **version**: Version of the event
- **eventType**: Type of the event
- **eventTime**: Time when the event was generated
- **severity**: Can be INFO or ERROR
- **eventSummary**: Brief description of the event
- **affectedEntities**:
- entityType: A string representing universal agent, UNIVERSAL_AGENT
- entityId: ID of the Universal Agent
- correlationKeys: Not applicable for Universal Agent
- properties: Not applicable for Universal Agent
- markedAsRead: Not applicable for Universal Agent
- markedAsResolved: Not applicable for Universal Agent
- archived: Not applicable for Universal Agent
- endTime: Not applicable for Universal Agent
- guid: Not applicable for Universal Agent
- subtype: Success or error code
- triggeredEntity:
  - entityType: A string representing universal agent: UNIVERSAL_AGENT
  - entityId: ID of the Universal Agent

Get Event Details

Get event details for all events. The number of events that can be returned is limited to 600.

```
GET /controller/universalagent/v1/user/events/details&timeRange=<value>
```

Query Parameters

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>timeRange</td>
<td>Time range specifier string as used in other parts of the Controller. For example:</td>
</tr>
<tr>
<td></td>
<td>• last_1_hour.BEFORE_NOW.-1.1424287621109.60</td>
</tr>
<tr>
<td></td>
<td>• last_5_minutes.BEFORE_NOW.-1.-1.5</td>
</tr>
</tbody>
</table>

Example Command

```
curl -i -X GET -u '<userName>@<accountName>:<password>'
-H 'Content-type: application/json'
http://<controller-host>:8080/controller/universalagent/v1/user/events/details?timeRange=last_5_minutes.BEFORE_NOW.-1.-1.5
```

Response Format
[{
"eventDetails": [
{
"id":<long>,
"version":<long>,
"name":<string>,
"value":<string>
}
]
}
],
userComments: [ ],
eventId: <long>
},...
]

Response Fields
- **eventId**: ID of the event
- **version**: Version of the event
- **name**: Details of the event
- **value**: Not applicable for Universal Agent

Get Event Details By Event ID
Get details for a specific event by event id.

GET /controller/universalagent/v1/user/events/detailsByEventId/<eventId>

Path Parameters

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Value</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>eventId</td>
<td>ID of the event</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Example Command
curl -i -X GET -u '<userName>@<accountName>:<password>'
   -H 'Content-type: application/json'
http://<controller-host>:8080/controller/universalagent/v1/user/events/detailsByEventId/1

Response Format

```json
{
   "eventDetails": [
   {
      "id":<long>,
      "version":<long>,
      "name":<string>,
      "value":<string>
   }
   ]
   eventProperties: [],
   userComments: [ ],
   eventId: <long>
}
```

Response Fields

- **eventDetails**: Not applicable for Universal Agent
  - *id*: ID of the event
  - *version*: Version of the event
  - *name*: Details of the event
  - *value*: Not applicable for Universal Agent
- **eventProperties**: Not applicable for Universal Agent
- **userComments**: Not applicable for Universal Agent
- **eventId**: Not applicable for Universal Agent

Create a Configuration Template

Create a configuration template to reference in your rulebooks using:

```
PUT /controller/universalagent/v1/user/configurations/{name}
```

The template is stored on the Controller and can be referenced from multiple rulebooks. See Rulebook Configuration Templates.

Path Parameters
<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Value</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the configuration template.</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Example Create Template**

```bash
curl -i -X PUT -u '<userName>@<accountName>:<password>' -H 'Content-type: application/json' -H 'Accept:application/json' http://localhost:8080/controller/universalagent/v1/user/configurations/440startedconfig -d '{
  "name": "450startedconfig",
  "config": {
    "state": "started",
    "version": "4.5.0"
  },
  "model": "NameValue"
}'
```

**Response Format**

Response Payload: Created template

```json
{
  "name": "450startedconfig",
  "model": "NameValue",
  "config": {
    "state": "started",
    "version": "4.5.0"
  },
  "referredRuleBooks": []
}
```

HTTP Response Status Code: 200 OK

Response Fields:

- name: Name of the configuration template
- model: Model of the template. NameValue is the only supported model.
- config: Configuration properties, a list of name-value pairs.
- referredRuleBooks: A list of the rulebooks that refer to this template.

**Update a Configuration Template**

Update an existing configuration template using:
PUT /controller/universalagent/v1/user/configurations/{name}

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Value</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the configuration template.</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Example Update Template Command

```bash
curl -i -X PUT -u '<userName>@<accountName>:<password>' -H 'Content-type: application/json' -H 'Accept:application/json'
http://localhost:8080/controller/universalagent/v1/user/configurations/450startedconfig -d ' {
  "name": "450startedconfig",
  "config": {
    "state": "started",
    "version": "4.5.0" # changed
  },
  "model": "NameValue"
}
```

Response Format

Response Payload: Changed template

HTTP Response Status Code: 200 OK

```
{
  "name": "450startedconfig",
  "model": "NameValue",
  "config": {
    "state": "started",
    "version": "4.5.0"
  },
  "referredRuleBooks": []
}
```

Response Fields

- **name**: Name of the configuration template
- **model**: Model of the template. NameValue is the only supported model.
- **config**: Configuration properties, a list of name-value pairs.
referredRuleBooks: A list of the rulebooks that refer to this template.

Delete a Configuration Template
Delete a configuration template using:

```
DELETE /controller/universalagent/v1/user/configurations/{name}
```

**Path Parameters**

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Value</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the configuration template.</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Example Delete Template Command**

```
curl -i -X DELETE -su
'\<controller-user>\:<controller-account>\:<controller-password>' \-H
"Content-type: application/json" \-H "Accept:application/json" \ 
"http://\<controller-host>\:<controller-port>/controller/universalagent/v1/ 
user/configurations/450startedconfig"
```

**Response**

Response Payload: None

HTTP Response Status Code: 200 OK, if the template is present, 404 Not Found, for non-existent templates.

Fields: Not Applicable

**View Configuration Templates**

View specific templates by name or view a list of all templates using:

```
GET /controller/universalagent/v1/user/configurations
GET /controller/universalagent/v1/user/configurations/{name}
```

**Path Parameters**

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Value</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of the configuration template.</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Example Command: View Specific Template

curl -i -X GET -su
"http://<controller-host>:<controller-port>/controller/universalagent/v1/user/configurations/450startedconfig"

Example Command: View All Templates

curl -i -X GET -su
"http://<controller-host>:<controller-port>/controller/universalagent/v1/user/configurations"

Response Payload for Specific Template

```json
{
    "name": "450startedconfig",
    "model": "NameValue",
    "config": {
        "state": "started",
        "version": "4.5.0"
    },
    "referredRuleBooks": []
}
```

Response Payload: All Templates

```json
{}  
```
HTTP Response Status Codes:

- 200 OK
- 404 Not Found: when template is not present.

Response Fields

- **name**: Name of the configuration template
- **model**: Model of the template. NameValue is the only supported model
- **config**: Configuration properties, a list of name-value pairs
- **referredRuleBooks**: A list of the rulebooks that refer to this template

For view all templates: List of configuration templates

[] if no configuration templates are present
Troubleshoot Universal Agent Setup

On this page:
- About Universal Agent Troubleshooting
- Troubleshooting Workflow

If you are having trouble setting up the Universal Agent, there are a few things to take a look at, as described in this topic.

About Universal Agent Troubleshooting

The Universal Agent log files can help you troubleshoot your Universal Agent configuration. These logs provide detailed information on the activities of the Universal Agent itself, but for a complete picture of the setup, they may not be the only resource you rely upon. Since the JVM has its own log context, you may need to check its logs to analyze unexpected behavior or results.

Troubleshooting Workflow

The general recommended workflow for troubleshooting Universal Agent setup issues is the following:

1. Missing or incorrect connection settings are a common agent misconfiguration issue. Check the rulebooks applied by the Universal Agent to make sure that the connection settings are correct. Verify the account name and key, along with the controller host and port.
2. Check the Universal Agent log file. The log files are located in the log directory for the monitor. This log file may be in one of the following locations:
   a. If the runtime_directory attribute is not specified in the Java Agent rule:
      `<universal_agent_home>/monitor/java/ua<version>/logs`
   b. If the runtime_directory attribute is specified in the Java Agent rule:
      `<runtime_directory>/logs`
      Where the exact location of runtime_directory is defined by the rule attribute.
3. Enrich the JVM log with Universal Agent logging by setting the following environmental variable in the JVM environment. Note that this requires a restart of the JVM:

   ```
   export APPDYNAMICS_UA_DEBUG=true
   ```
Sensitive Data Collection and Security

On this page:
- Preventing Sensitive Data Collection
- SaaS Deployment
- On-premises Deployment
- Role-based Access Control
- Suppress Raw SQL Capture
- Hide Query Literals
- Exclude Error Logs
- Mask Log Analytics Values
- Disable the Data Collector
- Filter Sensitive Data in Environment Variables
- Data Privacy Policy Dialog

AppDynamics collects data on the performance, health, and resources of an application, its components (transactions, code libraries), and the related infrastructure (nodes, tiers) that services those components. Before using AppDynamics, you should ensure that your application is secure and does not expose sensitive data.

Preventing Sensitive Data Collection

If your environment contains sensitive data that should not be processed by an AppDynamics product or sent to the AppDynamics SaaS environment, you should avoid the following:

- Applications that transmit sensitive data in URL query parameters
- Enabling HTTP request parameter capture
- Enabling bind variable capture
- Applications that send sensitive data in error logs and log files
- Allowing method invocation data collection
- Log captures. If you do capture logs, ensure that you mask values in those logs.

The following sections cover additional measures that you can take to ensure you do not expose sensitive data.

SaaS Deployment

AppDynamics supports encryption at rest in its SaaS deployments on personally identifiable information and sensitive business data.

On-premises Deployment

AppDynamics offers an on-premises solution for customers who want to maintain full control over their deployment of the software. With this type of implementation, AppDynamics has no access to the software or the data it collects and processes. Customers subject to strict regulatory requirements for data security may want to consider an on-premises solution. On-premises customers are responsible for encrypting their data by either using self-encrypting drives or other non-product solutions.

Role-based Access Control

You can use role-based access controls (RBAC) to limit the number of users who can access data collection features. The controls let you restrict a user's access to specific functions, data, analytics queries, and APIs.

You can control user access to data by specifying permissions for each user role. To configure user access, navigate to Settings > Administration. For more information, see Analytics and Data Security and Roles and Permissions.
Suppress Raw SQL Capture

Application Monitoring collects raw SQL as prepared statements captured with dynamic parameters bound to runtime values. You can disable the capture of raw SQL if it contains sensitive data. When you disable raw SQL capture, the SQL call appears in its original form, but with question mark parameters in place of sensitive data.

To disable the capture of raw SQL for an application, navigate to Configuration > Instrumentation > Call Graph Settings > SQL Capture Settings. Uncheck the Capture Raw SQL box.

You can also disable bind variable capture. Bind variables are placeholders for literal data in your SQL statements. When you disable bind variable capture, the values of bind variables are not displayed. For more information, see Call Graph Settings.

Hide Query Literals

Database Visibility hides query literals by default since queries can contain sensitive user data.

To verify that query literals are hidden for a database, navigate to Configuration. In the Security section, ensure that you have chosen the Remove literals from the queries option box. For more information, see Configure Query Literals Security.

You may also want to use bind variables as placeholders for literal data in your SQL statements.

Exclude Error Logs

Application Monitoring logs exceptions and errors that match parameters you specify in your custom logger. You may want to exclude sensitive payload data so that it does not show up in error logs.

To exclude a class in your application:

1. Navigate to Tiers & Nodes > Actions > Configure App Server Agent.
2. Select Use Custom Configuration.
3. Click the ( + ) button to create a new agent property.
4. Set the agent property name to exceptions-to-ignore.
5. Set the agent property value to the name of the class you want to exclude.

For more information, see Error Detection.

Mask Log Analytics Values

When configured, Application Analytics collects performance data from your app server agents, data from your log files, and performance and sessions data from End User Monitoring. You can mask sensitive information in your log analytics data.

To mask log analytics data:

1. Navigate to Analytics > Configuration > Log Analytics > Source Rules.
2. Click on the source rule that you want to specify masking for.
3. In the Field Management tab, next to ThreadName, you can specify the starting and ending position of the data you want to mask, and the character to use as the masking value.

For more information, see Configure Log Analytics Using Source Rules.

Disable the Data Collector

You can suppress data collection of HTTP request payloads, raw SQL, and other user data.

For the Java Agent, configure the disabled-features node property in the Controller UI.
For the .NET Agent, edit the config.xml file and set the disabled-features property to the names of features that you want to disable.

```xml
<property name="disabled-features"
value="RAW_SQL,LOG_PAYLOAD,METHOD_INV_DATA_COLLECTOR,HTTP_DATA_COLLECTOR,CUSTOM_EXIT_SNAP_DATA"/>
```

For more information, see App Agent Node Properties Reference and .NET Agent Configuration Properties.

**Filter Sensitive Data in Environment Variables**

You can mask sensitive data found in Java environment variables and system properties. To mask sensitive data, add the sensitive-data-filter property to app-agent-config.xml. The valid attributes are applies-to, match-type, and match-pattern. For more information, see Filter Sensitive Data.

**Data Privacy Policy Dialog**

Data collection has regulatory, legal, and customer-defined policies that you must follow. AppDynamics provides a data privacy policy reminder, in the form of a UI dialog, when you or your users configure parts of the AppDynamics products that could be used to collect regulated or other protected information.

This customizable statement is present in all areas of the AppDynamics UI where you can configure data collection. AppDynamics displays a default message if you have not made any customizations.

AppDynamics logs an event when it displays the data privacy policy dialog to you or another user.
Data Collection Dashboard

On this page:

- About the Data Collection Dashboard
- Data Collector Types

You can view all of the data collected and processed by the AppDynamics Application Performance Management (APM) Platform on the Data Collection Dashboard.

About the Data Collection Dashboard

As you may know, AppDynamics is very configurable, giving you control over most data configurations. However, due to the various AppDynamics agent types and products, you can change the configuration in many different places. This flexibility can make it challenging to keep track of all of your settings. The Data Collection Dashboard addresses this challenge by providing a single, easy-to-access view of the current state of all of your configuration parameters that may affect your organization's security and privacy. The Data Collection Dashboard should help you understand what data AppDynamics is collecting for your configured application instance at any point in time.

The unified view is only available to admins.

To view the Data Collection Dashboard, click the gear icon (⚙️) and then click Data Collection Dashboard.
The dashboard provides a read-only view. You can change the configuration values in the respective places of the platform.

You can find online help for a data collector type. To do so, navigate to Applications and choose an application. Then click Configuration > Instrumentation > Data Collectors. Click on help (?) in the data collector box of interest to view the online help for it. The online help contains more information on that particular data collector and how to configure it.

You can click Export to save a raw json file of the selected data collection settings, to share with non-admins, auditors, and sales.

Data Collector Types

This table lists all of the data collector types that you can view on the Data Collection Dashboard.

<table>
<thead>
<tr>
<th>Data Collector Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agents Logging Session</td>
<td>Agents Logging Sessions log information with specific type of logging information, such as application configuration changes, metric data, and system agent registrations.</td>
</tr>
<tr>
<td>Agents Metadata Default Config</td>
<td>Agents Metadata Default Configs collect app agents transaction metadata.</td>
</tr>
<tr>
<td>Configuration</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Browser Settings Config</td>
<td>Browser Settings Configs collect Browser RUM and Synthetic metrics.</td>
</tr>
<tr>
<td>Custom Data Browser</td>
<td>Custom Data Browsers customize the agent to collect custom metrics for the browser.</td>
</tr>
<tr>
<td>Custom Data Mobile</td>
<td>Custom Data Mobiles customize the agent to collect custom data types such as information points, custom timers, and custom metrics.</td>
</tr>
<tr>
<td>DB Collector</td>
<td>Database Collectors run within the Database Agent to collect performance metrics about your database instances and servers.</td>
</tr>
<tr>
<td>DB Security Config</td>
<td>Database Security Configs collect literals, which can contain sensitive user data, retrieved from the database.</td>
</tr>
<tr>
<td>EUM Settings Config</td>
<td>EUM Settings Configs collect EUM metrics, custom data, and Analytics data.</td>
</tr>
<tr>
<td>GUID Injection</td>
<td>GUID Injections correlate logs and transaction analytics data to specific business transaction requests.</td>
</tr>
<tr>
<td>HTTP Request</td>
<td>HTTP Requests capture the URLs, parameter values, headers, and cookies of HTTP messages exchanged in a business transaction.</td>
</tr>
<tr>
<td>Information Points</td>
<td>Information Points reflect the data state across all invocations of a method, independently of business transactions.</td>
</tr>
<tr>
<td>Log Analytics Source Rule</td>
<td>Log Analytics Source Rules log data sources using job files to collect log analytics data.</td>
</tr>
<tr>
<td>Method Invocation</td>
<td>Method Invocations capture code data, such as method arguments, variables, and return values.</td>
</tr>
<tr>
<td>Mobile Settings Config</td>
<td>Mobile Settings Configs collect Mobile RUM metrics.</td>
</tr>
<tr>
<td>SIM HTTP User Service Config</td>
<td>SIM HTTP User Service Configs collect Standalone Machine Agent data using HTTP listeners.</td>
</tr>
<tr>
<td>SQL Data Collector</td>
<td>SQL Data Collectors collect the business data from SQL parameters for use in transaction analytics.</td>
</tr>
</tbody>
</table>
Manage Users and Groups

Most Controller UI pages are access-controlled. After you install AppDynamics, you can add user accounts in the Controller UI, allowing other users to access the UI and configure AppDynamics. The Account Owner is the predefined role with the Administer users, groups, and roles permissions.

The Controller can authenticate users against local user accounts or external LDAP or SAML-based authentication providers. For information on setting up the Controller to use an external authentication provider, see User Management.

Users, Groups, and Roles

A user can belong to one or more groups. Groups let you assign and manage roles for users collectively.

Roles are an essential concept in the Controller UI. Roles determine what users can see or do in the UI, including which business applications they can monitor and the types of configuration changes they can make. Parts of the UI are not visible to users whose roles do not authorize access to those features. A user or group can have more than one role, but should have at least one.

AppDynamics comes with a set of predefined roles, but you can add your own, particularly to set up user access by business application. For more about roles, see Roles and Permissions.

Accessing Authentication Settings

You create and administer users in the Controller from the Administration page accessible from the gear icon ( ). You must be logged in as a user with the account owner role in the UI to see the Settings configuration options.

External authentication settings are configurable from the Authentication Provider tab in the Administration page. For more about setting up external authentication settings along with advanced options, see LDAP Authentication or SAML Authentication.

You can create an API Client from the API Clients tab in the Administration page. You can use the API Client to provide secure access to the Controller through REST API calls. For more information see API Clients.

Authentication settings in the Controller are specific to an account within the Controller. If you have a multi-tenant on-premises Controller, each account needs to be configured with authentication settings individually.

Creating Local Users

A local user is a user whose account credentials are stored in the Controller and who is authenticated by the Controller rather than by an external authentication provider. You can create local user accounts in the Users tab of the Administration page.

These guidelines apply to local user accounts:

- Because of browser incompatibilities, AppDynamics recommends using only ASCII characters for user names and passwords.
- Choose at least one role for the new user. If you do not choose a role before saving, a warning message appears in the UI. You can assign the user to a role later, but the user will not be able to use any features in the UI until assigned a role.

After creating a user, you can modify, delete, or duplicate the user account, or assign the user to a group or role from the users tab.

If deleted user owns a custom dashboard, then the dashboard and its associated shares and reports cease to function
As indicated in the UI, a user should have at least one role, which you can assign directly or through a group. Without a role, a user can log in, but will not be able to do much else in the Controller UI. You can associate users with roles from the user's configuration or in the Roles tab. Under Roles, the user and group assignments appear in the Users and Groups with this Role tab.

Be careful to avoid accidentally removing yourself from all groups or from all roles. Also, if the only roles of which you are a member are custom roles, do not delete those custom roles or remove permissions from them. Doing so can result in being locked out of the AppDynamics UI with no permissions at all. If this happens, use the built-in administrator role to restore the account.

### Require Strong User Passwords

As an account administrator, you can require local users (those authenticated by AppDynamics) to use strong passwords.

By default, strong password requirements are not enforced, which means that users can configure passwords of any length or complexity. To enforce strong password requirements, in the Administration page, open the Authentication Provider tab and select the Require Strong Passwords checkbox.

With the requirement enabled, passwords must meet the complexity requirements shown in the Authentication Provider tab of the Controller UI. The requirements include having at least eight characters, containing both upper and lower case letters, and more.

Passwords set by users after you enable this requirement must meet the requirements listed in the UI. Changing this option does not affect passwords that have already been set. That is, existing weak passwords will continue to work after you enable strong passwords.

### Create and Manage Groups

You can manage roles for local users collectively using groups in the Groups tab on the Administration page. If you are using LDAP to authenticate all AppDynamics Controller users you do not need to create AppDynamics groups.

After creating the group, assign users to the group by selecting the group and selecting the Member check boxes for the users to be added to the selected group or groups. Similarly, to associate the group to a role, select the Member check boxes for the roles to be associated with the selected group or groups.

You can associate groups with roles from the group configuration or under Roles in the Users and Groups with this Role tab.
Roles and Permissions

On this page:

- Predefined Roles
- Creating Custom Roles

Related pages:

- Account Permissions
- Application Permissions
- Business iQ Analytics Permissions
- Custom Dashboard Permissions
- Database Permissions
- End User Monitoring Permissions
- Server Visibility Permissions

Roles define a set of permissions that users of the AppDynamics Controller may have within the AppDynamics managed environment. This is also called role-based access control, or RBAC.

The Controller UI enables you to apply permissions at a fine-grained level. For example, you can grant permission to configure only a single application or a particular tier or to access a particular feature of the UI, such as custom dashboards.

Predefined Roles

The Controller UI includes predefined roles for administrator and read-only users. You cannot edit the predefined role permissions, however, you can create new roles as described in Creating Custom Roles.

The Controller UI includes these roles:

- Account Owner: Can add or edit users, groups, roles, and the authentication provider. This role has most of the account-level permissions and is sometimes known as the account administrator. See Account Permissions.
- Administrator: Can view and modify components that change state, such as applications, business transactions, dashboards, and so on. Can create War Rooms, view business flows, view and configure scheduled reports. This role can not add or edit users, groups, or roles.
- Analytics Administrator: Can view and grant access to Analytics features, such as creating API keys, creating metrics, creating extracted fields, and granting access for viewing analytics data. The Analytics Administrator has the capability to control which roles have access to specific applications or log source types. Additionally, the Analytics Administrator is the only user who is in charge of saved searches. By creating different saved searches, the Analytics admin can provide different data access levels to analytics users. For more details, see:
  - Analytics and Data Security
  - Business iQ Analytics Permissions
- Dashboards Viewer: Can view custom dashboards.
- DB Monitoring User: Can view the Database Monitoring UI. Cannot add, edit or delete database collectors.
- DB Monitoring Administrator: Can view the Database Monitoring UI and add, edit or delete database collectors.
- Server Monitoring Administrator: Can view the Server Monitoring UI and configure Service Monitoring features including Service Availability Monitoring.
- Universal Agent Administrator: Can configure and view the Universal Agent
- Universal Agent User: Can view the Universal Agent
- Applications and Dashboards Viewer: Can view all applications and their dashboards but cannot edit any. (formerly known as the Read-Only User)
- Workflow Executor: Can execute workflows.

To view the predefined roles

1. While logged in to the Controller UI as an Administrator or Account Owner, click the gear icon (⚙️) > Administration.
2. Click the Roles tab to view the list of predefined roles.
3. Click the **Users and Groups with this Role** tab to view users and groups assigned to a selected role.
Creating Custom Roles

Users with the Account Owner role or the Administer users, groups, roles ...permission can create new custom roles in the Controller UI. A common strategy for designing roles is to create a role with the minimum permissions allowable for all users, such as view permissions. Then you can create roles that use customizations of that minimum permission role to give additional, explicit permissions to a specific feature or business application.

You can clone predefined roles as a starting point for creating your own customized roles, but you should not assume the cloned roles have all of the permissions of the predefined role. In some cases, there may be hidden permissions, so you should add or remove permissions as needed for your customized role to ensure that you get the RBAC result you need.

To create or edit a custom role

a. While logged in to the Controller UI as an Account Owner, or other role with the Administer users, groups, roles ...permission, click the gear icon (⚙️) > Administration.
b. Click the Roles tab to view the list of predefined roles.
   From the tab, you can create new roles and modify or delete custom roles.
c. Click + Create to create a custom role.

d. Configure permissions by clicking the tabs.
   - Account
   - Applications
   - Databases
   - Analytics
   - Dashboards
   - Users and Groups with this Role
**Account Permissions**

You can set AppDynamics account permissions for custom roles from the Account tab in the Controller Administration UI. Most installations have one account per Controller. Usually, only very large installations or installations that have very distinct sets of users may require multiple accounts. Multiple accounts are part of the multi-tenant mode of installing the Controller. See [Controller Deployment](#) for details. In multi-tenant mode, the Controller UI context is divided into separate accounts. Each account has its own set of users, agents reporting to it, and application monitoring configuration.

The account predefined role is the **Account Owner**. This role is also known as the account administrator and has all account-level permissions except Execute Workflows. This role also has the following permissions:

- Can Create Applications and Dashboards
- All Database Visibility (Databases) related permissions

Most Account-level permissions can be considered administrative permissions and apply account-wide across business applications, products, and multiple application instances for the same account. The following table lists the permissions that can be set at the Account level.

<table>
<thead>
<tr>
<th>Permission</th>
<th>Activities enabled</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administer users, groups, roles, authentication, etc. View AppDynamics Agents. Use Agent Download Wizard</td>
<td>Add or edit users, groups, roles, and the authentication provider. Download agents using the download wizard.</td>
<td>Manage Users and Groups</td>
</tr>
<tr>
<td>View and Configure Licenses</td>
<td>View and create rules to specify the number of Application Performance Monitoring licenses to allocate to specified applications and machines.</td>
<td>License Management</td>
</tr>
<tr>
<td>Configure Email / SMS</td>
<td>Edit email and SMS settings used by AppDynamics to send alerts</td>
<td>Enable an Email Server</td>
</tr>
<tr>
<td>Configure Email Templates</td>
<td>Configure email templates for use in notification alerts.</td>
<td>Email Templates</td>
</tr>
<tr>
<td>Configure HTTP Request Templates</td>
<td>Configure the HTTP request templates used by HTTP request actions, which are triggered by AppDynamics policies.</td>
<td>HTTP Request Actions and Templates</td>
</tr>
<tr>
<td>Create War Rooms</td>
<td>Create (start) a war room. War rooms are collaborative custom dashboards created in real time.</td>
<td>Virtual War Rooms</td>
</tr>
<tr>
<td>View Business Flow</td>
<td>View all applications in a multi-business-application flow map, including those for which they are not granted explicit application permissions. Does not grant permission to drill down to applications that the user does not have permission to view. To see the downstream metrics and snapshots for the correlated application, the user must be a member of a role with view permissions to that business application.</td>
<td>Cross Application Flow</td>
</tr>
<tr>
<td>View Scheduled Reports</td>
<td>View scheduled reports from the Dashboards and Reports section of the UI.</td>
<td>Reports</td>
</tr>
<tr>
<td>Configure Scheduled Reports</td>
<td>Create, delete, send, or update scheduled reports.</td>
<td>Reports</td>
</tr>
<tr>
<td>View Universal Agent</td>
<td>Can use Universal Agent REST APIs to view information.</td>
<td>Universal Agent REST APIs</td>
</tr>
<tr>
<td>Configure Universal Agent</td>
<td>Can use Universal Agent REST APIs that add or change Universal Agent configuration.</td>
<td>Universal Agent REST APIs</td>
</tr>
</tbody>
</table>
Application Permissions

On this page:
- Create Default Permissions
- Customize Application Permissions
- Overlapping Role Permissions Examples
- General Permissions
- Application and Tier Permissions

Related pages:
- Roles and Permissions

You can set application permissions for custom roles from the Applications tab in the Controller Administration UI. You can assign the Can Create Applications permission to a custom role.

Application permissions follow an inheritance model. There are three levels in the model listed here in order from highest (default) to lowest (tier-specific):

- Default permissions
- Application-wide permissions
- Tier-specific permissions

By default, each level inherits from the one above it, unless you customize permissions at a lower level. This mechanism enables you to grant access to groups or users for specific business applications in the Controller UI.

Customized permissions at a specific level override more general permissions at another level. That is, tier-specific permissions take precedence over application-specific permissions, and application-specific permissions override default permissions. Not all permissions can be customized at the tier-level.

Create Default Permissions

All new applications inherit default permissions.

To configure default application permissions

1. From the Controller Administration UI, add or edit a custom role for which you want to grant default application permissions.
2. On the Applications tab, to grant the role permission to create new applications, click Can Create Applications.
3. Under Default Permissions, select the default permissions for this role: View, Edit or Delete.

- Check Delete to grant permissions to delete any application. To grant permission to delete a specific application, customize the permission at the application level.
- To grant specific permissions to edit specific application configurations for all applications:
  i. Click Edit to give all permissions to all applications or deselect Edit, and then click Edit(None).
ii. In the Edit Permissions window select the permissions for this role.

For information about the permissions that can be granted at the application level and tier levels, review the Application and Tier Level Permissions table.

4. Click OK in the Edit Permissions window.
5. Click Save at the top of the pane to save the configuration for this role.

Customize Application Permissions

To customize business application level permissions, follow these steps:

1. Choose Custom from the Permissions menu for the application (replacing the value of Inherited).
2. Check View option and then Edit(None), as shown in the following screenshots. You can also grant permission to delete a specific application.

To customize permissions at the tier level, expand the application permission tree and click Edit.
3. In the dialog box, choose the individual permissions for the selected tier and click OK.
4. Click Save when you are finished selecting permissions.

Overlapping Role Permissions Examples

Within specific and default permissions, granting a specific permission takes precedence over denying the same permission elsewhere. So, if a user is assigned two roles and one grants a permission and the second role denies it, the user will have permissions for the activity.

The following examples are designed to illustrate how overlapping permissions of different roles interact. The examples enable view, edit, and delete permissions to applications as shown for two Groups. The last column shows the resulting permissions for a specific user with roles that are assigned to each group.

<table>
<thead>
<tr>
<th>Group 1</th>
<th>Group 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Permissions (view, edit delete all applications)</td>
<td>Default Permissions (view, edit delete all applications)</td>
</tr>
<tr>
<td>Explicit permissions (view, edit delete application-1)</td>
<td>Explicit permissions (view, edit delete application-1)</td>
</tr>
<tr>
<td>A</td>
<td>None</td>
</tr>
<tr>
<td>B</td>
<td>Yes</td>
</tr>
<tr>
<td>C</td>
<td>Yes</td>
</tr>
</tbody>
</table>

- Result for example A: User has view, edit, and delete permissions to all applications, including application-1.
- Result for example B: User has view, edit, and delete permissions to all applications, excluding application-1.
- Result for example C: User has view, edit, and delete permissions to all applications, excluding application-1.

General Permissions

<table>
<thead>
<tr>
<th>Permission</th>
<th>Activities Enabled</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Can Create Applications</td>
<td>Create business, browser, and mobile applications. Also controls the Archive Snapshot action.</td>
<td>Business Applications</td>
</tr>
<tr>
<td>View, Edit and Delete permissions for new applications can be set as part of the default permissions for a custom role</td>
<td>View, edit or delete business applications (and the tiers and nodes), browser and mobile applications. Setting default delete permissions allows the user to delete all three artifacts from the application model.</td>
<td>Business Applications, Tiers and Nodes</td>
</tr>
</tbody>
</table>

Application and Tier Permissions

You can grant the following permissions as specified. Permissions that can be customized at the tier level are indicated in the
Description column. Asterisks (*) in the permissions table indicate permissions that are considered sensitive for security and data privacy purposes. Carefully consider the security and data privacy policies of your organization before granting these permissions.

<table>
<thead>
<tr>
<th>Permission</th>
<th>Description of Activities Enabled</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure Transaction Detection*</td>
<td>Create, edit, or delete transaction detection - can be at the tier level.</td>
<td>Transaction Detection Rules</td>
</tr>
<tr>
<td>Configure Backend Detection</td>
<td>Create, edit, or delete backends - can be done at tier level.</td>
<td>Backend Detection Rules</td>
</tr>
<tr>
<td>Configure Error Detection</td>
<td>Create, edit, or delete error detection.</td>
<td>Error Detection</td>
</tr>
<tr>
<td>Configure Diagnostic Data Collectors*</td>
<td>Create, edit, or delete diagnostic data collectors.</td>
<td>Data Collectors</td>
</tr>
<tr>
<td>Configure Call Graph Settings</td>
<td>• Edit call graph settings (no SQL) • Turn on or off capture raw SQL (call graph and SQL bind must both be on)</td>
<td>Call Graph Settings</td>
</tr>
<tr>
<td>Configure JMX</td>
<td>Create, edit, or delete JMX metrics.</td>
<td>Configure JMX Metrics from MBeans</td>
</tr>
<tr>
<td>Configure Memory Monitoring</td>
<td>Configure which custom classes are tracked by Object Instance Tracking. Note: To enable or disable Object Instance Tracking, you need the Configure Agent Properties permission.</td>
<td>Object Instance Tracking for Java</td>
</tr>
<tr>
<td>Configure EUM (for Browser RUM)</td>
<td>See End User Monitoring Permissions</td>
<td>Configure the Controller UI for Browser RUM</td>
</tr>
<tr>
<td>Configure EUM (for Mobile RUM)</td>
<td>See End User Monitoring Permissions</td>
<td>Configure the Controller UI for Mobile RUM</td>
</tr>
<tr>
<td>Configure Information Points*</td>
<td>Create, edit, or delete information points</td>
<td>Information Points</td>
</tr>
<tr>
<td>Configure Health Rules</td>
<td>Create, edit, or delete health rules</td>
<td>Configure Health Rules</td>
</tr>
<tr>
<td>Configure Actions</td>
<td>• Create, edit, or delete actions on agent properties UI • Create, edit, or delete email digests</td>
<td>Alert and Respond Actions Email Digests</td>
</tr>
<tr>
<td>Configure Policies</td>
<td>Create, edit, or delete policies.</td>
<td>Configure Policies</td>
</tr>
<tr>
<td>Configure Business Transactions</td>
<td>• Organize Business Transactions including: • Group business transactions • Exclude/un-exclude business transactions • Delete business transactions • Enable business transaction lockdown • Rename business transactions • Configure business transaction thresholds • Configure snapshot settings • Set as a background task • Configure data collectors • Enable End User Monitoring • Enable analytics for business transactions • Enable or disable GUID injection</td>
<td>Organize Business Transactions Transaction Thresholds Transaction Snapshots Monitor Background Tasks Data Collectors Set Up and Access Browser RUM Collect Transaction Analytics Data Business Transaction and Log Correlation</td>
</tr>
<tr>
<td>Feature</td>
<td>Description</td>
<td>Permissions</td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Configure Baselines</td>
<td>Create, edit, or delete baselines.</td>
<td>Dynamic Baselines</td>
</tr>
<tr>
<td>Configure SQL Bind Variables*</td>
<td>Turn on or off capture raw SQL (also requires Configure Call Graph Settings).</td>
<td>Call Graph Settings</td>
</tr>
</tbody>
</table>
| Configure Agent Properties                 | • Create, edit, or delete agent configuration (can be done at tier level).  
• Enable or disable automatic leak detection (can be done at tier level).  
• Enable or disable object instance tracking (can be done at tier level).  
• Enable or disable custom memory structure (can be done at tier level). | App Agent Node Properties  
Object Instance Tracking for Java  
Custom Memory Structures for Java |
| Agent Advanced Operation                   | • Reset agent from the node dashboard.  
• Request agent thread dumps.  
• Request agent debug logs.                                                                                                                                                                                   | Manage App Agents  
Diagnostic Actions  
Request Agent Log Files |
| Set JMX MBean Attributes and Invoke Operations | Edit MBean attributes or invoke actions on operations.                                                                                                                                                        | Monitor JMX |
| Configure Service Endpoints                | Create, edit, or delete service endpoints.                                                                                                                                                                  | Service Endpoint Detection |
| Configure Monitoring Level (Production/Deployment) | Switch between production and development mode.                                                                                                                                                           | Development Level Monitoring |
| Configure ‘My Dashboards’ for Tiers and Nodes | Create, edit or delete custom dashboards (can be done at tier level).                                                                                                                                       | Create and Manage Custom Dashboards and Templates  
Custom Dashboards |
| Create Events                              | Create, edit, or delete events.                                                                                                                                                                             | Alert and Respond API |
| Start Diagnostic Sessions                  | Start a diagnostic session.                                                                                                                                                                                  | Diagnostic Sessions |
| View Sensitive Data*                       | In combination with the Configure Transaction Detection permission, enables the use of Live Preview and Business Transaction Discovery features to stream live data from your application. | Custom Match Rule  
Live Preview |
Business iQ Analytics Permissions

You can set analytics permissions for custom roles from the Analytics tab in the Controller Administration UI. The tabs referred to here are part of the Administration UI and can be seen when creating a new role.

The Analytics predefined role is the Analytics Administrator. The following screenshot shows the analytics permissions categories. When creating a new role, select the permissions that you want to assign to the new role.

If a user does not have access to a specific application on the Applications tab but has access to Business Transactions for that application on the Analytics tab, they can do all actions for that application in the Analytics screens, except for creating metrics. For the user to create metrics from Analytics searches, you must assign the user to a role with view access to the Analytics Application on the Applications tab.

Asterisks (*) in the permissions tables on this page indicate permissions that should be considered sensitive for security and data privacy purposes. Carefully consider the security and data privacy policies of your organization before granting these permissions. For details on permissions related to Analytics data and functionality, see the links in the More Information column.

<table>
<thead>
<tr>
<th>Permission Name</th>
<th>Activities Enabled</th>
<th>For more information, see</th>
</tr>
</thead>
<tbody>
<tr>
<td>General &gt; Manage Fields*</td>
<td>Show and hide fields in analytics data so that you can restrict sensitive data to proper roles and users.</td>
<td>Managing Field Visibility</td>
</tr>
<tr>
<td>General &gt; Manage APIs*</td>
<td>View the analytics API tab to create and manage API authentication keys. Users with this API permission have access to all analytics data since they can use cURL with the access key and access all data.</td>
<td>Managing API Keys</td>
</tr>
<tr>
<td>General &gt; Manage Metrics</td>
<td>Create metrics from analytics searches. This permission controls which roles and users can create metrics from analytics searches. Once the metric exists, you can set up alerts in the usual way.</td>
<td>Create Analytics Metrics From Scheduled Queries</td>
</tr>
<tr>
<td>Search Permissions &gt; Can Create a Search*</td>
<td>Create and save an analytics search. Configure view, edit, and delete permissions for each saved search.</td>
<td>Analytics and Data Security</td>
</tr>
<tr>
<td>Search Permissions &gt; Saved Searches*</td>
<td>View, edit or delete a specific saved analytics search</td>
<td>Analytics and Data Security</td>
</tr>
<tr>
<td>Transaction Permissions*</td>
<td>View analytics transaction data from all applications or specific applications</td>
<td>Analytics and Data Security</td>
</tr>
<tr>
<td>Log Permissions*</td>
<td>View analytics data for all Source Types or specific log sources</td>
<td>Analytics and Data Security</td>
</tr>
<tr>
<td>Browser Requests and Sessions Permissions*</td>
<td>View analytics data from Browser requests</td>
<td>Analytics and Data Security</td>
</tr>
<tr>
<td>Mobile Requests and Sessions Permissions*</td>
<td>View analytics data from mobile requests and crash reports</td>
<td>Analytics and Data Security</td>
</tr>
<tr>
<td>Synthetic Sessions Permissions</td>
<td>View synthetic data from all or specific applications</td>
<td>Analytics and Data Security</td>
</tr>
<tr>
<td>Connected Devices Permissions</td>
<td>View analytics data for connected devices</td>
<td>IoT Monitoring</td>
</tr>
<tr>
<td>Custom Analytics Events Permissions*</td>
<td>Query custom analytics events data. Permissions can be granted to view data for all custom analytics events or on an event by event basis.</td>
<td>Analytics and Data Security</td>
</tr>
</tbody>
</table>
Custom Dashboard Permissions

You can set custom dashboard permissions for custom roles from the Dashboards tab in the Controller Administration UI.

The predefined role is the Dashboards Viewer. The permissions of this role are limited to viewing custom dashboards in the Controller UI. The permissions for this role apply if you do not set more specific permissions for an individual custom dashboard.

Each custom dashboard inherits the default custom dashboard permissions unless you override the defaults by configuring separate explicit permissions for individual dashboards.

For example, you could have a custom dashboard called SalesDashboard and a custom role SalesRole, and a second custom dashboard called FinanceDashboard and a second custom role FinanceRole. The SalesRole could be configured to have permissions in the SalesDashboard but not in the FinanceDashboard and so on.

Creating custom dashboard templates requires the Configure ‘My Dashboards’ for Tiers and Nodes permission, which you can set at the application level. See Application Permissions.

Default Permissions

Changes made to the default or custom permissions are automatically applied to existing and future dashboards. These permissions can be used to create special permissions for specific dashboards, and override default permissions.

To configure default permissions for a dashboard:

1. On the Administration UI under Roles tab, add or edit a custom role for which you want to grant default permissions.
2. Now click Dashboards tab and select Can Create Dashboards checkbox to grant role permission to create new dashboards.
3. Under Default Permissions, select the default permissions for this role: View, Edit or Delete.
4. Click Save at the top of the pane to save the configuration for this role.

Note
If any of the dashboard permissions in the custom permissions list matches with the default permissions, that entry will be removed from the custom list.
## Dashboard Permissions

<table>
<thead>
<tr>
<th>Permission</th>
<th>Activities Enabled</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>View</td>
<td>View specific custom dashboards</td>
<td>[Custom Dashboards]</td>
</tr>
<tr>
<td>Edit</td>
<td>Edit specific custom dashboards</td>
<td>[Custom Dashboards]</td>
</tr>
<tr>
<td>Delete</td>
<td>Delete specific custom dashboards</td>
<td>[Custom Dashboards]</td>
</tr>
<tr>
<td>Can Create Custom Dashboards</td>
<td>Create new custom dashboards</td>
<td>[Custom Dashboards]</td>
</tr>
</tbody>
</table>
Database Permissions

You can set Database permissions for custom roles from the Databases tab in the Controller Administration UI.

You can view metrics associated with database collectors, ensure that the Custom view permission is enabled for the Databases application listed under Administration >> Roles >> Applications. From the Databases tab, you can configure database collector permissions for custom roles.

For each custom role, you can select which databases the user is allowed to view, edit, or delete and you can enable global permissions applying to database collectors.

<table>
<thead>
<tr>
<th>Permission</th>
<th>Activities Enabled</th>
<th>For more information, see</th>
</tr>
</thead>
<tbody>
<tr>
<td>Can Configure Collectors</td>
<td>Configure baselines, wait state filtering, and custom metrics, as well as enable or disable the masking of query literals</td>
<td>Discover Normal Database and Server Activity Wait State Filtering Configuring Custom Metrics Configuring Query Literals Security</td>
</tr>
<tr>
<td>Can Create Collectors</td>
<td>Create database collectors for any database</td>
<td>Add Database Collectors</td>
</tr>
<tr>
<td>View</td>
<td>View all existing and new database collectors, and therefore the metrics for the databases associated with those collectors</td>
<td>Database Dashboard</td>
</tr>
<tr>
<td>Edit</td>
<td>Edit database collector fields for any database</td>
<td>Add Database Collectors</td>
</tr>
<tr>
<td>Delete</td>
<td>Delete database collectors for any database</td>
<td>Add Database Collectors</td>
</tr>
</tbody>
</table>
End User Monitoring Permissions

You can set End User Monitoring permissions for custom roles from the Applications tab in the Controller Administration UI. The Configure EUM permission is needed to configure Browser and Mobile RUM. You can add this permission as the default permission for all applications, or you can specify the permission at the application level.

Browser RUM

The Configure EUM permission permits the following actions for Browser RUM:

- Configure and download the JavaScript Agent
- Add include/exclude rules for base pages, Ajax requests, and virtual pages
- Enable JavaScript error capture and add rules to ignore specific JavaScript errors
- Set thresholds for slow end-user experience
- Enable the collection of slow, periodic, and error snapshots

See Configure the Controller UI for Browser RUM for instructions and a list of the supported Browser RUM configurations.

Browser Synthetic Monitoring

The Configure EUM permission permits the following actions for Browser Synthetic Monitoring:

- View, schedule, edit, and delete synthetic jobs

Mobile RUM

The Configure EUM permission permits the following actions for Mobile RUM:

- Upload dSYM files for iOS or ProGuard files for Android
- Add include/exclude rules for network requests and the Events Service
- Set thresholds for slow end-user experience
- Configure mobile crash alerts
- Configure mobile screenshots

See Configure the Controller UI for Mobile RUM for instructions and a list of the supported Mobile RUM configurations.
Server Visibility Permissions

You can assign the Server Visibility user permissions when you are creating custom roles. You can assign them as part of the default permissions for all applications or you can create customized applications permissions for new roles.

1. From the Controller Administration UI, navigate to Roles > Applications.
2. Select an existing role or create a new role.
3. In the Server Monitoring dropdown list, select Customized.
4. Click View and Edit to see the available permissions.

Predefined Roles

- Server Visibility Administrator
- Server Visibility User

Permissions

<table>
<thead>
<tr>
<th>Permission Name</th>
<th>Activities! Enabled</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure Server Visibility</td>
<td>Can view all Servers tabs and windows and configure Service Availability Monitoring. (<a href="#">Service Availability Monitoring</a>).</td>
</tr>
<tr>
<td>View Server Visibility</td>
<td>Can view all Servers tabs and windows. (<a href="#">Server Visibility</a>)</td>
</tr>
</tbody>
</table>
External Authentication Providers

Local user accounts are created and maintained within the Controller Admin UI. You can alternatively authenticate and authorize Controller UI users for your organization using external authentication providers.

Accessing Authentication Provider Settings

As an administrator in the Controller UI, you can view Authentication Provider settings in the **Settings > Administration** page.

Authentication Mechanisms

By default, Controller authentication is configured for AppDynamics, the local authentication option. Alternatively, you can have an external LDAP or SAML system perform Controller user authentication and authorization.

To view more information, see the following topics:

- LDAP Authentication
- SAML Authentication
This topic describes how to integrate AppDynamics Controller with LDAP directory servers.

About LDAP Support

You can delegate Controller UI authentication and authorization to external directory servers that comply with LDAP (Lightweight Directory Access Protocol) version 3.

While the Controller should be able to work with any LDAPv3-compliant server, the Controller has been verified to work with these LDAP products:

- Microsoft Active Directory for Windows Server 2008 SP2+
- OpenLDAP, 2.4+

To configure LDAP authentication in the Controller, you need to configure connection settings to the LDAP server and the queries that return user or group data. By mapping LDAP groups to roles, you can provision permissions in the AppDynamics Controller based on LDAP groups.

What Happens if the LDAP Server Becomes Unavailable?

If the LDAP server configured for Controller authentication becomes unavailable for any reason, the Controller falls back to local user authentication. Given this possibility, you should provision local user accounts in AppDynamics for the administrative users who will need access if the LDAP server becomes unavailable.

What Happens if a User Cannot be Found in the LDAP Directory?

If a user cannot be found in the LDAP directory, the authentication failure event is logged as a warning. The user, whether it is a regular controller user or a REST client user, may still be authenticated through local authentication.

Preparing the LDAP Directory for AppDynamics Integration

To use an LDAP authentication provider, your AppDynamics Controller needs to be able to connect to the external LDAP server. A good practice is to create a user account in LDAP specifically for the Controller to use to authenticate itself to the server and run the queries. The Controller user only needs to have search privileges in LDAP.

While you can map existing LDAP group definitions to roles in AppDynamics, your existing groups may not correspond directly to roles in AppDynamics. The easiest way to map LDAP groups to Controller roles is to create a group in LDAP for each role you want to be mapped in AppDynamics. LDAP groups for each role provides you with a manageable, 1-to-1 correspondence between your LDAP groups and AppDynamics roles.

For example, a possible LDAP group scheme for mapping in AppDynamics would be:

- AppDynamics-AppA-ReadOnly
- AppDynamics-AppA-Admins
- AppDynamics-AppA-DashboardViewers
- AppDynamics-AppB-ReadOnly
- AppDynamics-AppB-Admins
- AppDynamics-AppB-DashboardViewers

The sample group names imply having custom roles in AppDynamics targeted to specific applications, AppA and AppB.
Naming the groups with a common prefix, as the AppDynamics- prefix in our sample, allows you to use a relatively simple LDAP group filter. A group filter for the sample groups could be:

```bash
(&{objectClass=group}(cn=AppDynamics-*))
```

Using Paged Results for Large Result Sets

LDAP servers are sometimes configured to limit the number of entries it can return in a query response. If the results of your user or group query exceed that limit, AppDynamics reports a max_results_exceeded error.

To avoid this error, first try to refine your query filter to produce a smaller result set. The results must include the users who will need to access the AppDynamics UI.

If your LDAP server supports it, you can also enable paged results in the Controller LDAP configuration. With paged results, the LDAP server divides the result set into separately transmitted blocks.

The paged results feature applies to the behind-the-scenes interaction between the AppDynamics Controller and the backend LDAP server. It does not affect the UI view of the data.

LDAP Authentication with a SaaS AppDynamics Controller

Depending on your organization's security policies, it may not be possible to use LDAP authentication with the SaaS AppDynamics Controller, since doing so requires opening your firewall to permit Controller access to your corporate LDAP server.

However, if you do want to enable LDAP authentication with SaaS AppDynamics Controller, you will need to permit access through the firewall for the IP range of 69.27.44.0/24, the IP address range assigned to AppDynamics SaaS Controllers. The firewall rule should allow incoming LDAP requests from the Controller at the LDAP port you configure.

Before Starting

To perform LDAP configuration, you must have:

- An LDAP server. There is a one-to-one correspondence between an AppDynamics account and an LDAP server.
- An account on an AppDynamics SaaS or on-premises Controller
- Account administrator privileges on the AppDynamics Controller, as described in User Management.
- Network connectivity between your LDAP server and the Controller. If using a SaaS Controller, the LDAP server may not be accessible to the Controller without enabling access through your network firewall. See LDAP Authentication with a SaaS AppDynamics Controller.

Configuring LDAP Authentication

At a high level, the steps for setting up LDAP authentication include:

- Configure the connection to the LDAP server.
- Configure and test the LDAP query that returns users to be provisioned in the AppDynamics Controller.
- Configure the LDAP query that returns the LDAP groups to be mapped to AppDynamics roles.
- Map the users or groups to roles in AppDynamics.

Configure the Connection to the LDAP Server
As an administrator or account owner in the Controller UI, you can configure LDAP authentication from the Authentication Provider tab under Settings > Administration.

If the user or group query that you need to use will return more entries than permitted by the LDAP server and the server support paged results, configure paged results as follows:

- **Enable Paging**: Check this option to have the Controller request paged results from the server when submitting user or group queries.
- **Page Size**: Enter the number of entries per round-trip from the AppDynamics Controller to the LDAP server. The default is 500.

The page size should be the total number of entries to be returned divided by the number of round trips between the LDAP server and the Controller that are tolerable. For example, if you expect to receive 1200 results in a query and you can tolerate a maximum of two round trips, set the page size to 600 (1200 /2). See Using paged results for large result sets for more information.

Configure the LDAP connection settings:

- **Host**: Required address of the LDAP server.
- **Port**: Port on which the LDAP server listens. The default is 636 for an SSL connection and 389 if not using SSL. Required.
- **Use SSL**: Enabled by default to use a secure connection to the LDAP server. Clear if not using SSL.
- **Enable Referrals**: Enabled by default to support LDAP referrals. A referral is when an LDAP server forwards an LDAP client request to another LDAP server. Each referral event is referred to as a hop.
- **Maximum Referral Hops**: The maximum number of referrals that AppDynamics will follow in a sequence of referrals. The default is five.
- **Bind DN**: Distinguished Name of the user on the LDAP Server on whose behalf the AppDynamics application searches. Required.
- **Password**: Password of the user on the LDAP server. Required. Your settings should look something like this:

  Configure Users

  In the LDAP configuration page, configure information to find LDAP users:
• Base DN: Location in the LDAP tree to begin recursively searching for users. Required.
• Filter: Optional LDAP search string that filters the items matched from the base DN. See RFC22 54 for information about LDAP search filters.
• Login Attribute: The LDAP field that corresponds to the username users will enter when logging in to the AppDynamics UI. The default is **uid**. For Active Directory, this would typically be **sAMAccountName**.
• Display Name Attribute: The LDAP field to use as the user's display name.
• Group Membership Attribute: Optional user group membership field. Recommended for faster retrieval.
• Email Attribute: Optional user email address.

The Test Query button checks the connection. If successful, a screen displays the first few users returned by the query. (The test does not return the entire result set if the result set is large.)

**Configure Groups**

Optionally, you can map LDAP groups to user roles in the AppDynamics Controller. To do this, you need to set up the LDAP query that returns the LDAP groups to map, as follows.

• Base DN: Location in the LDAP tree to begin recursively searching for groups. Required.
• Enable Nested Groups: Option to include nested LDAP groups to a depth of 10.
• Filter: Optional LDAP search string that filters the items matched from the base DN. See RFC22 54 for information about LDAP search filters.
• Name Attribute: The LDAP field that contains the name of the group. Default is **cn**. Required.
• Description Attribute: The LDAP field that contains a description of the group. Optional.
• User Membership Attribute: Identifies members of the groups. Optional.
• Referenced User Attribute: Optional child attribute of the User Membership Attribute. Disabled if the parent is empty. Identifies property of the user that the user membership attribute contains.
The Test Query button checks the connection. If successful, the first few groups returned by the query are shown.

You can now assign permissions in the AppDynamics Controller to users or groups.

**Assign AppDynamics Permissions to an LDAP User**

1. In the Security Configuration window, click the **Users** tab. If LDAP is enabled and correctly configured, the AppDynamics Controller fetches the user names from the LDAP server.
2. Select the name of the user to whom you want to assign permissions.
3. In the Roles panel, check the roles that you want to assign to this user. You can assign multiple roles to a user.
4. Click **Save**.

**Assign AppDynamics Permissions to an LDAP Group**

LDAP Group configuration is optional.

1. In the Security Configuration window, click the **Groups** tab.
   - If LDAP is enabled and correctly configured, AppDynamics fetches the group names in LDAP.
2. Select the name of the group to which you want to assign permissions.
3. In the Roles panel, check the roles that you want to assign to this group. You can assign multiple roles to a group.
4. Click **Save**.

**Configuring the LDAP Cache Synchronization Frequency**

The Controller keeps information about LDAP users and groups in a local cache. It regularly connects to the LDAP server to synchronize its cache with the LDAP server.

The Controller caches information about users and group membership. It does not cache user passwords. Accordingly, the Controller authenticates the user credentials against the LDAP server at the start of every user session.

If a user account is removed from LDAP, the change is reflected immediately: that is, the user will not be able to log in to the Controller UI from that point. However, if the user has an existing session in the Controller UI, that session continues until the user logs out or the session expires.

If the user's access to the Controller is based on group membership and the user is removed from the group but maintains an account in the LDAP server, the user will be able to log in to the Controller until the next time synchronization with the LDAP server occurs. By the default synchronization frequency setting, this ability to access the Controller UI could continue for up to an hour.

You can modify the default synchronization frequency of one hour as described in the following procedures.
Configure the LDAP Synchronization Frequency

1. Stop the Controller application server:
   - On Linux, run:
     ```bash
     platform-admin.sh stop-controller-appserver
     ```
   - On Windows, run this command from an elevated command prompt, which you can open by right-clicking the Command Prompt icon in the Windows Start menu and choosing **Run as administrator**:
     ```cmd
     platform-admin.exe cli stop-controller-appserver
     ```

2. Open the `<Controller-Installation-Directory>/appserver/glassfish/domains/domain1/config/domain.xml` file for editing.

3. In the `<jvm-options>` element, add a system property named `appdynamics.ldap.sync.frequency` with the desired synchronization frequency in milliseconds.
   For example, to have the Controller synchronize to the LDAP server every 15 minutes (900000 milliseconds), add:
   ```xml
   <jvm-options>-Dappdynamics.ldap.sync.frequency=900000</jvm-options>
   ```
   The default is 3600000 milliseconds (1 hour).

4. Save the file.

5. Restart the Controller app server:
   - On Linux, run:
     ```bash
     platform-admin.sh start-controller-appserver
     ```
   - On Windows, run the following in an elevated command prompt:
     ```cmd
     platform-admin.exe cli start-controller-appserver
     ```
SAML Authentication

On this page:
- How SAML Authentication Works
- Sample SAML Request
- About Roles and SAML Groups
- Enabling SAML Authentication
- Configuring Default Permissions
- Mapping SAML Group to Roles
- Disabling SAML Authentication
- Troubleshooting

Related pages:
- Configure SAML for Microsoft Active Directory Federation Services 3.0
- Configure SAML for Microsoft Active Directory Federation Services 2.0 or 2.1
- Configure SAML for Okta
- Configure SAML for OneLogin

The AppDynamics Controller can use an external Security Assertion Markup Language (SAML) identity provider to authenticate and authorize users. This topic describes how to set up and administer SAML authentication.

After upgrading the Controller to 4.5.x, you may encounter issues where the SAML authentication request fails for accounts that use ADFS SAML.

We suggest you configure your ADFS to send the Name Id attribute in the SAML assertion. You can also create a shared local user and use the local login option to bypass the login issue.

How SAML Authentication Works

With SAML authentication enabled, the Controller UI redirects credentials entered in the login page to the external SAML identity provider. To be able to log in to the Controller UI, the user needs to be able to access both the Controller and the identity provider service by the network from their computer.

Roles govern user privileges in the Controller UI. For more information on roles, see Roles and Permissions. You can configure the Controller to assign roles to authenticated users based on group attributes in their SAML responses. See About Roles and SAML Groups for more information on mapping SAML attribute to roles.

When SAML authentication is enabled, users can authenticate with local, AppDynamics credentials by clicking the Use Local Login link. The link appears at the bottom of the Login page. Local authentication is useful if you haven’t mapped a particular role to SAML attributes, such as AppDynamics administrators, or if you need to disable SAML authentication.

Sample SAML Request

The SAML request that the external identity provider receives from the Controller looks something like the following:
About Roles and SAML Groups

The Controller can assign roles to SAML-authenticated users using one of the following mechanisms:

- **SAML group attributes**: You can map SAML group membership attributes to roles in AppDynamics. Using this method, each time the user authenticates, the Controller checks the SAML assertion and updates the role assignment if needed.
- **Internal AppDynamics account roles**: If a SAML-authenticated user has the same username as an AppDynamics internal user account and the SAML assertion does not contain mapped SAML group attributes, the Controller gives the user the roles for the internal AppDynamics account.
- **Default role**: If there are no SAML group attributes in a user’s identity assertion, the authenticated user is assigned the SAML default role upon the first log in. An AppDynamics administrator can verify and adjust the roles for users manually in AppDynamics once those users have accounts. Manually adjusted roles are preserved across subsequent log-ins.

To use SAML group attributes as the basis for AppDynamics role assignments, configure the SAML group attribute value mapping. If using internal account role associations, you can enable SAML authentication and configure basic SAML authentication settings.

The default role is not associated with any AppDynamics roles out-of-the-box, so you need to configure the default role to use it.

Enabling SAML Authentication

The following steps assume that you have an account with a supported identity provider. You need to know the SAML Login URL and have the x.509 certificate supplied by your identity provider. You should also be familiar with the format of the SAML identity response from your SAML provider.

The SAML authentication can handle encrypted SAML responses from identity providers. You can select which encryption algorithms to use while configuring your identity provider instances, such as AES128_CBC and AES256_CBC. To use AES256_CBC, you will have to perform extra configurations to the JRE to support this algorithm. Otherwise, the Controller will fail to decrypt the response.

To enable SAML encryption, make sure to select SAML Encryption on Authentication Provider of Administration on your Controller. You can then add the SAML Encryption Certificate and the SAML Encryption Key.

Enable SAML authentication as follows:

1. As a user with AppDynamics account administrator privileges in the Controller UI, click the gear icon (⚙️) > Administration.
2. Click on the Authentication Provider tab and select SAML as the authentication provider.
3. Enter the following SAML Configuration settings:
   - **Login URL**: The SAML Login URL where the Controller will route Service Provider (SP)-initiated login requests. This login URL is required.
   - **Logout URL**: The URL where the Controller will redirect users after they log out. If you do not specify a logout URL,
users will get the AppDynamics login screen when they log out.

- Certificate: The x.509 certificate from your identity provider configuration. Paste the certificate between the BEGIN CERTIFICATE and END CERTIFICATE delimiters. Avoid duplicating “BEGIN CERTIFICATE” and “END CERTIFICATE” delimiters from the source certificate itself.

4. In the SAML Attribute Mappings settings, specify how SAML-authenticated users are identified in the AppDynamics Controller as follows:

- Username Attribute: Unique identifier for the user in the SAML response. This value must be unique among all SAML users in the Controller account. Given the sample response below, the value for this setting would be `User.OpenIDName`.
- Display Name Attribute: The informal name for the user corresponding to the AppDynamics Name field. Given the sample response, this value would be `User.fullName`.
- Email Attribute: The user's email address, corresponding to AppDynamics email field. Given the sample response, this value would be `User.email`.
- Account Name: If the Controller is in multi-tenant mode, the SAML response must contain a custom SAML attribute `accountName` that indicates the user's AppDynamics account name. You cannot change this field mapping in the Controller.

If you choose to use encrypted assertions, you cannot simultaneously rely on passing your account name as an attribute in the SAML response. If you would like your SAML responses to be encrypted, you must include the query parameter `accountName` in the HTTP request. You can do this by setting the account name value in all of the connector apps, such as Okta, OneLogin, and AD.

```xml
<saml:AttributeStatement>
  <saml:Attribute Name="User.OpenIDName"
    NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
    <saml:AttributeValue
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:type="xs:string">adynamo</saml:AttributeValue>
  </saml:Attribute>
  ...
  <saml:Attribute Name="User.email"
    NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
    <saml:AttributeValue
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:type="xs:string">Ajay.Dynamo@example.com</saml:AttributeValue>
  </saml:Attribute>
  <saml:Attribute Name="User.fullName"
    NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
    <saml:AttributeValue
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:type="xs:string">Mr. Ajay Dynamo</saml:AttributeValue>
  </saml:Attribute>
  ...
</saml:AttributeStatement>
```

5. To map SAML group attributes to AppDynamics roles, configure the SAML Group Mappings settings. The settings you use depends on the structure of the SAML group attribute in the response, as described in Map SAML Groups to AppDynamics Roles. If you are using internal AppDynamics accounts to map user roles, you can skip this step.

6. Optionally specify a master SAML Access Attribute included in the SAML response from your provider. When enabled, the Controller grants access to users only when the SAML assertion contains a corresponding value for the attribute. In the sample
response below, this attribute value is AccessControl.

```xml
<saml:Attribute
    NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic"
    Name="AccessControl">
    <saml:AttributeValue
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:type="xs:string">{access}</saml:AttributeValue>
</saml:Attribute>
```

7. Click **Save** to apply your changes. The Controller immediately starts using the SAML identity provider you configured for user authentication.

**Configuring Default Permissions**

Instead of mapping SAML attributes to roles, you can allow all users to get a default role with the permissions you specify.

To use default permissions, edit the **Default Permissions** settings in the SAML Group Mappings list.

In the Default Group Mapping dialog, choose the AppDynamics roles that all authenticated users get.

**Mapping SAML Group to Roles**

If the identity assertion from the SAML provider includes group attributes that correspond to AppDynamics roles, you can configure mappings between those attributes and roles. The SAML Group Mappings settings in the SAML configuration page control the mappings, as described here.

To configure SAML attribute to role mapping:

1. In the SAML Group Attribute Name field, enter the Name attribute value that identifies the SAML Attribute element with group affiliations for the user. For example, given the following response snippet, use SAML groups-Membership in the SAML Group Attribute Name field.

```xml
<saml:Attribute
    NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic"
    Name="Groups-Membership">
    <saml:AttributeValue
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:type="xs:string">
        {group1};{group2}
    </saml:AttributeValue>
</saml:Attribute>
```

2. Use the Group Attribute Value and Mapping of Group to Roles settings to describe the structure of the SAML group attribute from which AppDynamics needs to extract the group value and the roles associated with those values. The Controller can extract Group Attribute values based on the following options:

- **Singular Group Values**: The response contains an `AttributeValue` element with a single group-mapping value.
• **Multiple Nested Group Values**: The response contains more than one `AttributeValue` element, each with a single group-mapping value.

• **Singular Delimited Group Value**: The response contains a single `AttributeValue` element with multiple, delimiter-separated group-mapping values.

• **Regex on Singular Group Value**: The response contains a single `AttributeValue` element from which you want to extract the group-mapping value with a regular expression.

The sections below provide more information on and examples for each option.

3. With any options selected, select the **Value is in LDAP Format** checkbox if the value or values returned by the group attribute value is in LDAP format. For example: "OU=AppDynamics-Users". With this option enabled, only "AppDynamics-Users" is used to map to the SAML Group name.

The following sections describe the SAML group attribute value mapping options.

### Singular Group Values

Choose **Singular Group Value** if the SAML group attribute contains a single group, as in the following example.

```xml
<saml:AttributeStatement>
  <saml:Attribute Name="Groups-Membership"
    NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
    <saml:AttributeValue
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:type="xs:string">Admin</saml:AttributeValue>
  </saml:Attribute>
</saml:AttributeStatement>
```

For this example, AppDynamics would extract the value Admin and associate the user with a SAML Group with the same name. In the following sample configuration, the user would get the roles configured assigned to the Admin SAML group in the example in the following figure, such as Account Administrator and Analytics Administrator.

### Multiple Nested Group Values

With this option selected, AppDynamics expects multiple `AttributeValue` child elements under the SAML Attribute with the group information, as in the following example:
SAML Group Attribute Name: Groups-Membership

Group Attribute Value:

- Singular Group Value
- Multiple Nested Group Values
- Singular Delimited Group Value
- Regex on Singular Group Value
- Value is in LDAP Format

Mapping of Group to Roles:

<table>
<thead>
<tr>
<th>SAML Group</th>
<th>AppDynamics Roles</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>GuestUser</em></td>
<td>User, Dashboard Viewer</td>
</tr>
<tr>
<td><em>DBManager</em></td>
<td>DB Monitoring Administrator, DB Monitoring User, User, Dashboard Viewer</td>
</tr>
<tr>
<td><em>Admin</em></td>
<td>Account Administrator, Analytics Administrator, Administrator, Server Monitoring Administrator, User, Server...</td>
</tr>
</tbody>
</table>


**Singular Delimited Group Value**

With this option selected, AppDynamics expects a single `AttributeValue` element with multiple, delimiter-separated values, as in the following example:

```
<saml:Attribute Name="Groups-Membership"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
  <saml:AttributeValue
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="xs:string">Admin;DB-Manager</saml:AttributeValue>
</saml:Attribute>
```
Specify the delimiter that separates the values to extract, such as a semi-colon in the example.

Given the following sample configuration, the user would get the AppDynamics roles associated with both the Admin and DB-Manager groups, such as the Dashboard Viewer, User, and DB Monitoring Administrator.

### SAML Group Mappings

<table>
<thead>
<tr>
<th>SAML Group Attribute Name</th>
<th>Groups-Membership</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group Attribute Value</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Regular Expression on Singular Group Value

Choose this option to have AppDynamics extract group mapping values using a regular expression. Regular expressions enable you to pull group values from unstructured contexts, such as from within a larger string, as in the following response example:

```xml
<saml:AttributeStatement>
  <saml:Attribute Name="Groups-Membership"
  NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
    <saml:AttributeValue
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:type="xs:string">User memberships in _Admin_ and _DBManager_ groups.</saml:AttributeValue>
  </saml:Attribute>
</saml:AttributeStatement>
```

In this example, the group names _Admin_ and _DBManager_ are embedded in the AttributeValue string. To extract those names, you can use a regular expression such as `_ [a-zA-Z]_`. Like other types of group attribute sources, AppDynamics assigns all roles associated with both the _Admin_ and _DBManager_ SAML Groups, as follows:
Disabling SAML Authentication

To disable SAML authentication, log in as an administrator using the local login link on the login page and restore the default authentication mode, AppDynamics authentication.

Troubleshooting

*Azure Environments*

When creating usernames in SAML in Azure environments, the UserName attribute is used if it is passed along. The NameID attribute is not handled if it is passed back to Azure. If you relied on the legacy behavior of concatenating the first name and surname, then you can workaround this enhancement by changing your Azure provider to return the concatenated names in the UserName attribute.
User Preferences

Controller UI users can change their passwords, account settings, date and time format, and other preferences in the My Preferences page.

User Preferences

You can access the controls for changing your display name, contact email, or password from the My Preferences page by clicking the Edit link under your email. The controls require you to enter your password to complete the change.

The display name is the name that the Controller uses to identify you in specific screen text and messages. For example, it appears in notifications to other Controller users when you share a dashboard with them. You cannot change your username. To change your username, you need to have an administrator delete your account and create another one with the new name.

You can display non-English languages in the Controller UI. To do so, enable your font preference to use system fonts. Using system fonts directs the UI to display text in the font configured for the system running the browser.

Some user account settings are attributes of local users (that is, users validated against credentials stored in AppDynamics). If your Controller is configured to use an external authentication mechanism instead, such as SAML or LDAP, you need to change the equivalent settings in the external system instead.

Strong Passwords

Your Controller administrator may require strong passwords for local accounts. A secure password must meet these requirements:

- At least eight characters in length
- Contain both uppercase and lowercase letters
- Include at least one number
- Not the same as your username or email address

View Preferences

The Controller UI provides the following user preferences:

- Date Format: By default, the format is MM/DD/YY (for example, 09/25/14). Choose an alternate format from the drop-down menu.
- Use 24 hour Time Format: Enable this option if you want the UI to represent time in 24-hour time format instead of 12-hour clock format.
- Enable Help Pop-ups: Help popups provide help text in context in the Controller UI. By default, they are enabled. To prevent help popups from appearing in the UI, clear this checkbox. Alternatively, you can prevent individual popups by selecting the Don't Show Again checkbox when the popup appears. To clear the list of popups marked as Don't Show Again, click the Reset All button.
- Graph Color Scheme for the Metric Browser: Select either Light or Dark to change the metric browser color scheme.
- Graph Color Scheme for All Other Graphs: Select either Light or Dark to change the navigation panel color scheme.
- Font: Determines the font type used in the UI. For screen text, the Controller UI uses a font set that it embeds by default. If the operating system of the computer on which you access the Controller UI uses a non-English language, you can configure the UI to use non-English languages by setting the font to use system fonts instead.
- Mouse Wheel Legacy Mode: If scrolling in the Controller UI using your mouse scroll does not work correctly, you should try enabling the Mouse Wheel Legacy Mode option. Enabling this option is may be necessary if accessing the Controller UI with certain older browsers.

You may need to log out of the UI and log back in to see the effects of your changes.
Advanced User Preferences

The user preferences page presents advanced options that include debug mode.

Debug Mode

The debug mode in the Controller UI is primarily intended for internal use by the AppDynamics development team. In some cases, you may be asked to enable debug mode when consulting with AppDynamics Support, for example, when you are troubleshooting an issue. However, it is important to note that specific debug mode options can negatively impact Controller performance. For this reason, you should only enable debug mode when directly advised to do so by AppDynamics Support.
Metrics and Graphs

On this page:
- About Metric Visualization Tools
- About Metrics

AppDynamics reports metrics for monitored systems over time.

The AppDynamics UI provides a variety of graphs and tools to help you visualize metric data so you can use it effectively for analyzing your application's performance, as described in this section.

**About Metric Visualization Tools**

Metric data appears throughout the UI. However, the Metric Browser is the main interface through which you inspect metrics. The Metric Browser displays all the metric values collected over the selected time period as an expandable tree.

In the browser, you can drag and drop metrics from the tree onto graphs to compare metrics and analyze patterns.

Analysis tools include:

- The scalability analysis tool provides scatter diagram graphs to display how performance scales as load (calls per minute) changes. You can compare load to response time or to CPU time for the entire application, a business transaction or a server.
- The correlation analysis tool provides scatter diagram graphs to compare any two metrics. You configure which metrics to display on the X and Y axes.

**About Metrics**

AppDynamics includes a wealth of metrics that reflect your application's performance. It also lets you extend and adapt metrics in specific ways. Information points, for example, allow you to create metrics based on data collected from application values, such as method parameters or return values. Percentile metrics let you configure metrics at deviation points that are useful to you.

You can create a metric from any application parameter using custom metrics. See Extensions and Custom Metrics for information on custom metrics.
The Metric Browser gives you a metrics-oriented view of the monitored environment for the time period selected in the time range menu.

In the Metric Browser, you can see metrics by nodes, business transactions, the overall application, and more. You can also compare two or more metrics and view metrics relative to a baseline. In general, the Metric Browser is the tool you use when you want drill into and analyze the details for a specific metric.

The following steps show how to use the Metric Browser:

1. Click **Metric Browser** in the left navigation pane.
2. In the browser, double-click on metrics in the left pane to graph data for that metric in the current time frame in the right pane. Add multiple metrics to compare them.
3. In the graph pane, you can choose the type of data you want to see for each metric, including observed values, minimum, maximum, sum, count, and the baseline.
4. From a filter, select if you want to view data from nodes that are currently receiving performance data, nodes that are not receiving performance data, or all nodes.
5. Other view options you can enable include the plot points on the graph line or adapting the scale of the y-axis unit based on the values in the graph, as in the following screenshot:

To remove metrics from the graph, click the x icon that appears at the right or click **Clear** at the top of the metric browser to remove all metrics.
Metric Data Point Details

For most types of metrics in the browser, you can click any of the points in the graph to view more information about the metric observed at that point in time. The information shown includes the metric identifier, date and time of the observation, along with any of the following values relevant to the metric:

- **Obs**: The observed average of all data points seen for that interval. For the Percentile Metric for the App Agent for Java, this is the percentile value. For a cluster or a time rollup, this represents the weighted average across nodes or over time.
- **Min**: The minimum data point value seen for that interval
- **Max**: The maximum data point value seen for that interval
- **Sum**: The sum of all data point values seen for that interval. For the Percentile Metric for the App Agent for Java, this is the result of the percentile value multiplied by the Count.
- **Count**: The number of observations aggregated in that one point. For example, a count of five indicates that there were five one-minute data points aggregated into one point.
- **Base**: The baseline data for the metric. For more information, see **Viewing Metrics Relative to Baselines**.

By selecting the check box that corresponds to the metric information at the bottom of the graph, such as Min, Max, Count, or Sum, you can graph that information for the metric over the time range shown in the browser. Check boxes for only the types of information relevant to the metric can be selected.

Min and max values are not available for any count- or sum-based metric except when the metric is rolled up to hourly or daily data points. Count- and sum-based metrics include errors per minute, calls per minute, and so on.

Analyze Performance for a Time Range

Once you identify a time frame of interest in the metric browser, you may want to see snapshots in that time range. To analyze snapshots for a time range or zoom in to a time range in the browser, click and hold your mouse or pointing device button at the start of the time range in the graph and drag the pointer to the end of the time range. When you release the button, the time range menu appears.
Viewing Metrics Relative to Baselines

You can see how data in the Metric Browser compares to baseline values. Monitoring baseline deviation is a good way to be aware of performance metrics that might be violating your Service Level Agreements (SLAs). For information about how baselines are calculated, see Dynamic Baselines.

To display baseline patterns, select the baseline option for the metric in the view options for the metric:

Exporting Metric Data

You can export metric data that is currently displayed to a comma-separated values (CSV) file. Exported data includes all the data displayed, even if there is a time range selected. If you want to export data only for a specific time range, first set a global time range and then export.
To export all data:

1. Choose **Actions** from the top menu of the browser, and then **Export Data**.
2. In the pop-up window, you can either copy the data to your clipboard or click **Export Data** to download the data as a CSV file.

To export data for an individual metric in the graph, click the download icon that appears to the right of the metric view options at the bottom of the graph.
Metric Data Resolution over Time

You can view AppDynamics metrics for different time intervals, or resolutions.

Metric Generation and Time Zone Differences

In a distributed system, the metrics that the Controller collects may come from different time zones.

Before sending metrics to the Controller, agents normalize the times associated with the metrics by applying a UTC (Coordinated Universal Time) offset appropriate for their region.

In the Controller UI, all times are displayed in the Controller user's local time. To render local times, the UI applies the offset needed to render UTC time into the user's time zone as configured on the computer on which the browser is running.

While the time zone used in the UI is taken from the local environment and cannot be modified apart from changing the computer's system time, each UI user can adjust the format of the time display, as described in User Preferences.

While the Controller UI displays times in the local system time, specific timestamps generated by AppDynamics reflect the Controller system time. These include, for example, event timestamps in notification messages, which indicate the Controller system time. Note that SaaS Controllers use Pacific Time (PT) as their system time.

Viewing Rolled Up Data in Graphs

Depending on your environment, AppDynamics measures hundreds, thousands, or even tens of thousands of metric values each minute. To display these results over time in a meaningful way, AppDynamics rolls up the data at regular intervals (for example, at one-minute, ten-minute, and one-hour intervals) based on how long ago the metric was measured.

In graphs throughout the AppDynamics user interface, you can hover over a point on the chart to get the value of the metric data measured during the interval represented by the point. The point represents the beginning of the interval. The length of the interval depends on the selected time range. Default settings are listed below.

- For up to 4 hours, data is rolled up every minute and displayed at 1-minute resolution. Each point on the graph represents a 1-minute interval; the point represents the beginning of the interval. Therefore, the data presented for a 5:00 AM point represents the values measured from 5:00 AM to the millisecond before 5:01 AM.
- After 4 hours, data is rolled up every 10 minutes and displayed at 10-minute resolution. Each point on the graph represents a 10-minute interval; the point represents the beginning of the 10-minute interval. Therefore, the data displayed for a 5:00 AM point represents the values measured from 5:00 AM to the millisecond before 5:10 AM.
- After 48 hours, data is rolled up every hour, and displayed at 1-hour resolution. Each point on the graph represents a 1-hour interval; the point represents the beginning of the hour. Therefore, the data displayed for a 5:00 AM point represents the values measured from 5:00 AM to the millisecond before 6:00 AM.
- After 365 days, data at 1-hour resolution is deleted.

Your administrator can adjust the amount of time data is retained at each resolution, although we do not generally recommend keeping data for longer than these default values.

The following sections discuss the roll-up periods and other time-related considerations when viewing metrics.

One-Minute Resolution

If you view data on a graph for any period in the previous four hours, each value you see represents a roll-up of all the values measured during that minute. For example, if AppDynamics measures the response time for an application for 300 calls in a minute, the response time values of those 300 calls are averaged to present the Average Response Time (ART) for that minute. When you hover over a point
on the chart, the Count figure represents how many times the metric data was measured during that minute.

In the graph that follows, response time was measured 140 times between 2:57 and 2:58 PM and the ART for that minute was 889 ms.

In the following chart, the time range was set for the last three hours, so the interval is one minute. The errors per minute value displayed at the 1:34 PM point represents the number of errors recorded during the minute from 1:34 PM to the millisecond just before 1:35 PM.

Ten-Minute Resolution

From 4 to 48 hours, metric data is displayed at a 10-minute resolution. If you view data in a graph for a period that is between 4 and 48 hours old, each value you see represents a roll-up of the values measured during that 10-minute period.

The ART for the rolled-up time period represents the true average. That is, the ART aggregates the sum and count of the total requests for the period; it does not average the ARTs calculated for the previous roll-up granularity.

One-Hour Resolution

After 48 hours, metric data is displayed at a 1-hour resolution. If you view data in a graph for a period that is more than 48 hours old, each value you see represents a roll-up of the values that were measured during that hour.
How the Last Interval is Displayed in a Chart

Because of the way rolled-up data is displayed, it may look as if data is not included for the last increment in the time range. However, the last data point, in fact, represents the last interval in the range.

For example, when you set a time range from 8 AM to 12 PM for a day that has been rolled up into 1-hour data points, you might expect to see five data points, one for 8 AM, 9 AM, 10 AM, 11 AM, and 12 PM. However, the returned data consists of four data points for the hours 8 AM, 9 AM, 10 AM, and 11 AM. The first data point, 8:00 AM, represents data collected from 8 AM to the millisecond just before 9 AM. The last data point, 11:00 AM, represents data collected from 11 AM to the millisecond just before 12 PM. So you are actually seeing data from 8 AM to 12 PM, even though the last data point in the graph is 11 AM.

Viewing Charts for Long Time Ranges

For time ranges that are greater than three weeks, graphs in the UI such as the Slow Transaction graph show the 14th of each month as the time point on the x-axis. For a time range that is less than two full months, this means that the graph may only have a single point, the middle of the month encompassed by the time range.

Viewing Details about Older Data

If you need to review details for an issue that occurred during a period for which you have only 10-minute or 1-hour data, AppDynamics provides access to diagnostic data via Transaction Snapshots. For example, suppose you were seeing values for 2 AM and 3 AM three days ago, but you need to examine details about a problem you were alerted about that occurred at 2:15 AM that morning. You view Transaction Snapshots for the hour between 2 AM and 3 AM to see details about what happened during that time period. By default, snapshots are retained for two weeks, and individual snapshots can be archived. See Transaction Snapshots.

If you are viewing a graph in the Metric Browser, you can select a specific period in the chart and drill down to see Transaction Snapshots and other information. See Analyzing Performance for a Specific Time Range.
Percentile Metrics

On this page:

- Disable Percentile Metric Collection
- Specify Percentiles to Collect
- Modify the Percentile Metric Algorithm

Related pages:

- Quick Tour of Percentile Metrics for Java

Works with:

Java  .net

A percentile is a value below which a given percentage of measurements in a set falls. For example, a 95th percentile value of 150 ms means that 95% of all values are 150 ms or less. This page describes how to modify default percentile collection.

Disable Percentile Metric Collection

The Java Agent and the .NET Agent capture percentile metrics by default. You can disable percentile metric collection in the Configure Percentile Metrics panel on the Configuration > Slow Transaction Thresholds window. Alternatively, manually set the disable-percentile-metrics node property to True to prevent the agent from collecting percentile metrics.

Specify Percentiles to Collect

By default, app agents capture the 95th percentile. You can indicate 5 whole numbers between 1 and 99 as Percentiles to Collect on the Configuration > Slow Transaction Thresholds window. You can apply the configuration changes to all existing business transactions or only to new transactions discovered after the configuration change.

Modify the Percentile Metric Algorithm

The agent uses one of the following methods to calculate percentile metrics:

- P Square algorithm (default): This option consumes the least amount of storage and incurs the least amount of CPU overhead. The accuracy of the percentile calculated varies depending on the nature of the distribution of the response times. You should use this option unless you doubt the accuracy of the percentiles presented.
- Quantile Digest algorithm: This option consumes slightly more storage and CPU overhead for the machine where the agent is running, but may offer better percentiles depending on how the response times are distributed.

To change the algorithm the agent uses to calculate percentiles set the percentile-method-option app agent node property.
Monitor Events

In the AppDynamics model, events represent a change in the state of a monitored application. An event can represent an error or exception generated by the application, the crossing of a performance threshold, or an operational change in the application, such as a JVM restart.

You can define policies that generate notifications or perform actions when an event occurs. See Alert and Respond for more information. You can also use events to investigate application issues underlying performance problems, as described here.

Viewing and Monitoring Events

You can view and analyze events in the Controller UI. From the Database Visibility, Server Visibility, or user experience application pages in the UI, you can view events by clicking Events from the left navigation tree.

The UI shows a subset of the types of events generated in AppDynamics. You can access additional event types via the REST API. See AppDynamics APIs for more information about internal events.

To see events for application monitoring, access a dashboard such as a business application, business transaction, tier or node dashboard and select the Events tab. The tab shows the events generated in the selected time frame.

In the Events list, you can filter the types of events displayed by clicking Filters and selecting the type of events to show.

You can view more information about an event from the list. If the event is an application change, such as an application restart, the details contain a description of the event. If the event is a slow transaction or an error, the details are transaction snapshots from which you can drill down to the root cause of the problem.

From the event details dialog, you can perform actions such as opening a War Room, delete or archive the event or test action execution.
Archiving Events

An event is purged after two weeks, at which point it is no longer accessible. To save an event beyond the default snapshot lifespan—for example, if you want to make sure an event associated with a particular problem is retained for future analysis—you can archive the event.

For on-premises Controllers, administrators can modify the default two-week period by configuring the `events.retention.period` property in the Controller Settings section of the Administration console.

To archive an event:

1. Select the event from the events list.
2. Click Actions > Archive Event.
3. Click Archive to confirm the action.

The archive icon ( ) in the right column indicates that an event has been archived. To view all archived events in a long list of events, you can click the column heading to sort by archiving. This groups the archived events together.

Filter Custom Events

The AppDynamics REST API enables you to define custom events. You can then filter on those events by creating a custom filter, as follows:

1. Open the filter options on the Events page.
2. Find the Filter by Custom Events section at the bottom of the panel, and click the add icon ( ).
3. In the Add Custom Event Filter dialog, enter the name of the custom event type.
4. Optionally, specify filtering event properties as key/value pairs. The custom event needs to contain the property values you specify to satisfy the filter. If you add multiple properties, a match value of All applies an AND operator to the list, meaning all properties need to match for the filter to be satisfied. Any indicates a logical OR, meaning at least one property must exist and match for the filter to be satisfied.
Monitor Application Change Events

When investigating an application error or poor performance, you may want to see whether any application changes occurred at the same time as the error or issue.

AppDynamics automatically detects and reports common application change events, such as app server restarts or changes to application environment variables. You can supplement automatically detected change events with custom application change events, generated either through the UI or the REST API.

Application changes appear alongside other events in the events list. Like other types of events, you can view details for the event by double-clicking the event or see the details in the context of other performance indicators, as the following describes. They can also serve as triggering conditions for policies.

Manually Registering Application Changes

You can add an application change event to AppDynamics manually.

This capability allows you to log a change event, in effect, which might not otherwise be detected by AppDynamics. Controller users can then correlate the event to performance data later. For example, you may want to create an event when you push an updated version of the application.

To create an application change event manually

1. In the Events List, expand the Actions menu and choose Register Application Change Event.
2. In the Register Application Change dialog, configure the event. You can choose the type of event, the scope, and the time when the event should be registered.

---

**Register Application Change Event**

- **Description**: Rolling out v2 update to Acme App
- **Type**: Application Deployment
- **What is the scope of this change?**: Application Wide
- **Change Date/Time**: Now
3. Click **Register Application Change Event**. If Now is selected as the change time, the event appears in the events list immediately:

4. Double-click the event in the list to view more details about the event, including any actions triggered by the event as a result of a policy.

**Automatically Registering Application Changes**

The AppDynamics **REST API** provides an alternative to creating application change events in the UI.

The API lets you automate event registration so that external systems can generate events based on their activities or conditions. For example, you can configure your release management system to generate an event in AppDynamics when it deploys an update to the application code, allowing you to correlate that event with application performance.

For information on creating a custom event via the REST API, see the **Create a Custom Event** method in **AppDynamics APIs**.

**Correlating Application Changes With Other Events**

You can view application changes in the context with other events and performance indicators in various Controller UI pages. The following steps describe how to view them in the transaction score histogram.

**To view application changes with other events in Transaction Score:**

1. In the Application Dashboard, click the **Transaction Score** tab.
2. Scroll down to the **Events** histogram.
3. Click **Add Criteria**.
4. From the dropdown, select **Event Types**.
5. Unselect all of the criteria except **Application Changes**.

Application change events appear in the **Events** histogram, where they appear in the context of other performance indicators.
Events Reference

On this page:

- ACTIVITY_TRACE
- ADJUDICATION_CANCELLED
- AGENT_ADD_BLACKLIST_REG_LIMIT_REACHED
- AGENTASYNC_ADD_REG_LIMIT_REACHED
- AGENT_CONFIGURATION_ERROR
- APPLICATION_CRASH
- AGENT_DIAGNOSTICS
- AGENT_ERROR_ADD_REG_LIMIT_REACHED
- AGENT_EVENT
- AGENT_METRIC_BLACKLIST_REG_LIMIT_REACHED
- AGENT_METRIC_REG_LIMIT_REACHED
- AGENT_STATUS
- ALREADY_ADJUDICATED
- APPDYNAMICS_DATA
- APPDYNAMICS_INTERNAL_DIAGNOSTICS
- APPLICATION_CONFIG_CHANGE
- APPLICATION_DEPLOYMENT
- APPLICATION_DISCOVERED
- APPLICATION_ERROR
- APP_SERVER_RESTART
- AZURE_AUTO_SCALING
- BACKEND_DISCOVERED
- BT_DISCOVERED
- BUSINESS_ERROR
- CLR_CRASH
- CONTROLLER_AGENT_VERSION_INCOMPATIBILITY
- CONTROLLERASYNC_ADD_REG_LIMIT_REACHED
- CONTROLLER_COLLECTIONS_ADD_REG_LIMIT_REACHED
- CONTROLLER_ERROR_ADD_REG_LIMIT_REACHED
- CONTROLLER_EVENT_UPLOAD_LIMIT_REACHED
- CONTROLLER_MEMORY_ADD_REG_LIMIT_REACHED
- CONTROLLER_METADATA_REGISTRATION_LIMIT_REACHED
- CONTROLLER_METRIC_DATA_BUFFER_OVERFLOW
- CONTROLLER_METRIC_REG_LIMIT_REACHED
- CONTROLLER_PSD_UPLOAD_LIMIT_REACHED
- CONTROLLER_RSD_UPLOAD_LIMIT_REACHED
- CONTROLLER_SEP_ADD_REG_LIMIT_REACHED
- CONTROLLER_STACKTRACE_ADD_REG_LIMIT_REACHED
- CONTROLLER_TRACKED_OBJECT_ADD_REG_LIMIT_REACHED
- CUSTOM
- CUSTOM_ACTION_END
- CUSTOM_ACTION_FAILED
- CUSTOM_ACTION_STARTED
- CUSTOM_EMAIL_ACTION_END
- CUSTOM_EMAIL_ACTION_FAILED
- CUSTOM_EMAIL_ACTION_STARTED
- DB_SERVER_PARAMETER_CHANGE
- DEADLOCK
- DEV_MODE_CONFIG_UPDATE
- DIAGNOSTIC_SESSION
- DISK_SPACE
- EMAIL_ACTION_FAILED
- EMAIL_SENT
- EUM_CLOUD_BROWSER_EVENT
- EUM_CLOUD_SYNTHETIC_BROWSER_EVENT
- EUM_INTERNAL_ERROR
This topic lists many of the types of events generated in AppDynamics. Specifically, it covers these types of events:

- Event types that are common to the platform, such as UI, configuration, and licensing-related events
- Events generated by the alert and respond system
- Application monitoring events

Information on event types that are specific to a platform module, if available, appears with the module-specific documentation. For example, Database Agent events are listed on Database Agent Events Reference.

Event types listed here that are not visible in the UI can be retrieved using the AppDynamics REST API.

**ACTIVITY_TRACE**
Description: The agent sent an internal event containing activity traces. Activity traces enable the tracing of a code path that passes through a specified class/method. The App Agent for Java uses them to provide object instance tracking (OIT) and automatic leak detection (ALD).

Category: AppDynamics Data

Visible in UI: No

Learn More: Java Memory Leaks, Object Instance Tracking for Java

ADJUDICATION_CANCELLED

Description: A designated approver has canceled a thread dump or remediation action that was triggered by a policy.

Visible in UI: No

Learn More: Actions Requiring Approval, Policies

AGENT_ADD_BLACKLIST_REG_LIMIT_REACHED

Description: The agent Application Diagnostic Data (ADD) blacklist registration limit has been reached.

Visible in UI: No

Learn More: Customize System Notifications

AGENT_ASYNC_ADD_REG_LIMIT_REACHED

Description: The agent async Application Diagnostic Data (ADD) registration limit has been reached.

Visible in UI: No

AGENT_CONFIGURATION_ERROR

Description: An agent configuration error has been detected.

Category: AppDynamics Configuration Warnings

UI Display name: Agent Configuration Error

Visible in UI: Yes

Learn More: Resolve .NET Agent Installation and Configuration Issues

APPLICATION_CRASH

Description: A crash has been detected for a JVM. The crash log file is updated.

Category: Application Changes

UI Display name: Application Crash

Visible in UI: Yes

AGENT_DIAGNOSTICS

Description: Diagnostic information concerning agent activity, such as business transaction overflow or HTTP error code diagnostics, has been sent.

Category: AppDynamics Data

UI Display name: Agent Diagnostics

Visible in UI: No
AGENT_ERROR_ADD_REG_LIMIT_REACHED

**Description:** The agent error Application Diagnostic Data (ADD) registration limit has been reached.
**Visible in UI:** No

AGENT_EVENT

**Description:** Generic internal event. This event type is also used for all Database Agent events with a case-specific message attached.
**Category:** AppDynamics Internal Diagnostics
**UI Display name:** Agent Event
**Visible in UI:** No
**Learn More:** Database Agent Events Reference

AGENT_METRIC_BLACKLIST_REG_LIMIT_REACHED

**Description:** The agent metric blacklist registration limit has been reached.
**Visible in UI:** No
**Learn More:** Metrics Limits, Customize System Notifications

AGENT_METRIC_REG_LIMIT_REACHED

**Description:** The agent metric registration limit has been reached.
**Visible in UI:** No
**Learn More:** Metrics Limits, Customize System Notifications

AGENT_STATUS

**Description:** The agent has been enabled or disabled.
**Category:** AppDynamics Internal Diagnostics
**UI Display name:** Agent Enabled / Disabled
**Visible in UI:** No
**Learn More:** Manage App Agents

ALREADY_ADJUDICATED

**Description:** A designated approver has previously approved or canceled a thread dump or remediation action that was triggered by a policy.
**Visible in UI:** No
**Learn More:** Actions Requiring Approval, Policies

APPDYNAMICS_DATA

**Description:** Events used to send data from the agent to the Controller, then to the UI. Events such as BTOverflowDetails and MemoryLeakDiagnostics should be moved into this bucket.
**Category:** AppDynamics Data
**Visible in UI:** No
APPDYNAMICS_INTERNAL_DIAGNOSTICS

Description: Internal diagnostics events.
Category: AppDynamics Internal Diagnostics
Visible in UI: No

APPLICATION_CONFIG_CHANGE

Description: Application configuration has been changed interactively by the user or through the REST API.
Category: Application Changes
UI Display name: Application Configuration Change
Visible in UI: Yes

APPLICATION_DEPLOYMENT

Description: An application has been deployed.
Category: Application Changes
UI Display name: Application Deployment
Visible in UI: Yes

APPLICATION_DISCOVERED

Description: A new application has been added to the Controller.
Category: Discovery
UI Display name: New Application Discovered
Visible in UI: Yes
Learn More: Policies

APPLICATION_ERROR

Description: An application error has been detected.
Category: Errors
UI Display name: Application Server Exception
Visible in UI: Yes

APP_SERVER_RESTART

Description: An application server has been restarted.
Category: Application Changes
UI Display name: App Server Restart
Visible in UI: Yes

AZURE_AUTO_SCALING

Description: An internal event that reports Azure auto-scaling progress to the UI. Seen at the bottom of the Azure auto-scaling screen.
Category: AppDynamics Data
Visible in UI: No

BACKEND_DISCOVERED
Description: A new backend has been added to the application.
Category: Discovery
UI Display name: New Backend Discovered
Visible in UI: Yes
Learn More: Policies

BT_DISCOVERED
Description: A new business transaction has been added to the application.
Category: Discovery
UI Display name: New Business Transaction Discovered
Visible in UI: Yes
Learn More: Policies

BUSINESS_ERROR
Description: A business error has been detected.
Category: Errors
UI Display name: Business Error
Visible in UI: Yes

CLR_CRASH
Description: A CLR crash has occurred.
Category: AD Infrastructure events
UI Display name: CLR Crash
Visible in UI: Yes

CONTROLLER_AGENT_VERSION_INCOMPATIBILITY
Description: The agent version is newer than the Controller version.
Category: AppDynamics Configuration Warnings
UI Display name: Agent Version is newer than Controller version
Visible in UI: Yes
Learn More: Agent and Controller Compatibility, Install the Controller

CONTROLLER_ASYNC_ADD_REG_LIMIT_REACHED
Description: The Controller limit for registering async Application Diagnostic Data (ADDs) for this account has been reached
Visible in UI: No
Learn More: Customize System Notifications

CONTROLLER_COLLECTIONS_ADD_REG_LIMIT_REACHED
Description: The Controller COLLECTIONS ADD registration limit for the account has been reached.
Visible in UI: No
Learn More: Customize System Notifications

CONTROLLER_ERROR_ADD_REG_LIMIT_REACHED
Description: The limit for registering error Application Diagnostic Data (ADDs) for the account has been reached.
Visible in UI: No
Learn More: Customize System Notifications

CONTROLLER_EVENT_UPLOAD_LIMIT_REACHED
Description: The limit on the number of events per minute that can be uploaded to the controller for the account has been reached.
Visible in UI: No
Learn More: Customize System Notifications

CONTROLLER_MEMORY_ADD_REG_LIMIT_REACHED
Description: The Controller MEMORY ADD registration limit for the account has been reached.
Visible in UI: No
Learn More: Metrics Limits, Customize System Notifications

CONTROLLER_METADATA_REGISTRATION_LIMIT_REACHED
Description: The Controller metadata registration limit for the account has been reached.
Visible in UI: No
Learn More: Business Transactions, Organize Business Transactions, Customize System Notifications

CONTROLLER_METRIC_DATA_BUFFER_OVERFLOW
Description: The Controller metric data buffer has overflowed. Now dropping metric data.
Learn More: Metrics Limits, Customize System Notifications

CONTROLLER_METRIC_REG_LIMIT_REACHED
Description: The limit for registering metrics for the account has been reached.
Visible in UI: No
Learn More: Metrics Limits, Customize System Notifications

CONTROLLER_PSD_UPLOAD_LIMIT_REACHED
Description: The PSD limit for the account has been reached.
Visible in UI: No
Learn More: Customize System Notifications

CONTROLLER_RSD_UPLOAD_LIMIT_REACHED
Description: The request segment data (RSD) limit for the account has been reached.
Visible in UI: No
Learn More: Customize System Notifications

CONTROLLER_SEP_ADD_REG_LIMIT_REACHED
Description: The limit for the Controller SERVICE ENDPOINT ADD registration for the account has been reached.
Visible in UI: No
Learn More: Customize System Notifications

CONTROLLER_STACKTRACE_ADD_REG_LIMIT_REACHED
Description: The limit for registering StackTrace ADDs for the account has been reached.
Visible in UI: No
Learn More: Customize System Notifications

CONTROLLER_TRACKED_OBJECT_ADD_REG_LIMIT_REACHED
Description: The Controller TRACKED_OBJECT ADD registration limit for the account has been reached.
Visible in UI: No
Learn More: Customize System Notifications

CUSTOM
Description: These are custom events thrown by REST API calls or Machine Agent API calls.
Category: Custom Event
UI Display name: Depends on the event.
Visible in UI: Depends on the event.
Learn More: Create Events

CUSTOM_ACTION_END
Description: A custom action has ended.
Visible in UI: No
Learn More: Custom Actions, Policies

CUSTOM_ACTION_FAILED
Description: A custom action has failed.
Visible in UI: No
Learn More: Custom Actions, Policies
CUSTOM_ACTION_STARTED

Description: A custom action has started.
Visible in UI: No
Learn More: Custom Actions, Policies

CUSTOM_EMAIL_ACTION_END

Description: A custom email action ended.
Visible in UI: No
Learn More: Diagnostic Actions, Policies

CUSTOM_EMAIL_ACTION_FAILED

Description: A custom email action failed.
Visible in UI: No
Learn More: Diagnostic Actions, Policies

CUSTOM_EMAIL_ACTION_STARTED

Description: A custom email action started.
Visible in UI: No
Learn More: Diagnostic Actions, Policies

DB_SERVER_PARAMETER_CHANGE

Description: The DBMS server parameters have been changed.
Category: DB Agent Event

DEADLOCK

Description: The agent has detected code deadlock.
Category: Code Problems
UI Display name: Code Deadlock
Visible in UI: Yes
Learn More: Code Deadlocks for Java

DEV_MODE_CONFIG_UPDATE

Description: The Dev Mode Config has been updated. This is fired from the agent and the Controller.

DIAGNOSTIC_SESSION

Description: A diagnostic session has started.
Category: AppDynamics Internal Diagnostics
UI Display name: Diagnostic Session
Visible in UI: No
Learn More: Diagnostic Sessions

DISK_SPACE
Description: The controller is running out of disk space.
Category: AppDynamics Configuration Warnings
UI Display name: Controller Disk Space Low
Visible in UI: Yes
Learn More: Controller System Requirements, Controller Disk Space and the Database, Database Size and Data Retention

EMAIL_ACTION_FAILED
Description: Email action has failed.
Learn More: Notification Actions

EMAIL_SENT
Description: Email was sent to notify the recipient of an event.
Visible in UI: No
Learn More: Notification Actions

EUM_CLOUD_BROWSER_EVENT
Description: A browser snapshot was stored in the database.
Visible in UI: No
Learn More: End User Monitoring, Browser Snapshots

EUM_CLOUD_SYNTHETIC_BROWSER_EVENT
Description: A synthetic browser snapshot was stored in the database.
Visible in UI: No
Learn More: End User Monitoring, Browser Snapshots

EUM_INTERNAL_ERROR
Description: An internal EUM error has occurred.
Visible in UI: No
Learn More: End User Monitoring

HTTP_REQUEST_ACTION_END
Description: An HTTP request action ended.
Visible in UI: No
Learn More: Diagnostic Actions, Policies
HTTP_REQUEST_ACTION_FAILED

Description: An HTTP request action failed.
Visible in UI: No
Learn More: Diagnostic Actions, Policies

HTTP_REQUEST_ACTION_STARTED

Description: An HTTP request action started.
Visible in UI: No
Learn More: Diagnostic Actions, Policies

INFO_INSTRUMENTATION_VISIBILITY

Description: Information was written to the Bytecode Transformer Log. This log contains information associated with the AppDynamics bytecode instrumentation (BCI) engine.
Category: AppDynamics Internal Diagnostics
UI Display name: Bytecode Transformer Log
Visible in UI: No
Learn More: Request Agent Log Files, App Agent Node Properties Reference

INTERNAL_UI_EVENT

Description: These are the XResponder.handleGeneralServerFaultEvent events.
Category: AppDynamics Internal Diagnostics
UI Display name: Controller API Call Threw an Exception
Visible in UI: No

LICENSE

Description: The AppDynamics license has expired.
Category: AppDynamics Configuration Warnings
UI Display name: License Expired
Visible in UI: Yes
Learn More: License Information

MACHINE_AGENT_LOG

Description: The Machine Agent is now logging information.
Category: AppDynamics Data

MACHINE_DISCOVERED

Description: A new machine has been added to the application.
Category: Discovery
UI Display name: New Machine Discovered
Visible in UI: Yes
Learn More: Policies

MEMORY
Description: Events for automatic leak detection and custom memory structures.
Category: AppDynamics Data
UI Display name: AppDynamics Data
Visible in UI: No

MEMORY_LEAK_DIAGNOSTICS
Description: The agent sends this internal event with memory leak data. The UI uses this on the memory monitoring screens in the node dashboards.
Visible in UI: No

MOBILE_CRASH_IOS_EVENT
Description: An iOS mobile application crash has arrived at the Controller.
Visible in UI: No
Learn More: Crashes

MOBILE_CRASH_ANDROID_EVENT
Description: An Android mobile application crash has arrived at the Controller.
Visible in UI: No
Learn More: Crashes

NETWORK
Description: The log data provided by the NPM Agent has been logged by the NPM Dynamic service.
UI Display name: Network
Visible in UI: Yes

NODE_DISCOVERED
Description: A new node has been added to the application.
Category: Discovery
UI Display name: New Node Discovered
Visible in UI: Yes
Learn More: Policies

NORMAL
Description: A business transaction is normal (not slow, very slow or stalled).
Visible in UI: No
Learn More: Transaction Thresholds, Dynamic Baselines

OBJECT_CONTENT_SUMMARY
Description: The agent sent an internal event with object content summary for collections, caches, etc.
Category: AppDynamics Data
Visible in UI: No

POLICY_CANCELED_CRITICAL
Description: A health rule violation, based on a status of critical, was canceled.
Category: Policy Violations
UI Display name: Health Rule Violation Canceled - Critical
Visible in UI: Yes
Learn More: Health Rules, Policies

POLICY_CANCELED_WARNING
Description: A health rule violation, based on a status of warning, was canceled.
Category: Policy Violations
UI Display name: Health Rule Violation Canceled - Warning
Visible in UI: Yes
Learn More: Health Rules, Policies

POLICY_CLOSE_CRITICAL
Description: A health rule violation, based on a status of critical, ended.
Category: Policy Violations
UI Display name: Health Rule Violation Ended - Critical
Visible in UI: Yes
Learn More: Health Rules, Policies

POLICY_CLOSE_WARNING
Description: A health rule violation, based on a status of warning, ended.
Category: Policy Violations
UI Display name: Health Rule Violation Ended - Warning
Visible in UI: Yes
Learn More: Health Rules, Policies

POLICY_CONTINUES_CRITICAL
Description: After the initial POLICY_OPEN_CRITICAL event, an event generated to indicate the continuation of the health rule violation at the critical level.
Category: Policy Violations
**UI Display name**: Health Rule Violation Continues - Critical  
*Visible in UI*: Yes  
*Learn More*: Health Rules, Policies

**POLICY_CONTINUES_WARNING**

**Description**: After the initial POLICY_OPEN_WARNING event, an event generated to indicate the continuation of the health rule violation at the warning level.

*Category*: Policy Violations  
*UI Display name*: Health Rule Violation Continues - Warning  
*Visible in UI*: Yes  
*Learn More*: Health Rules, Policies

**POLICY_DOWNGRADED**

**Description**: A health rule violation was downgraded from critical to warning.

*Category*: Policy Violations  
*UI Display name*: Health Rule Violation Downgraded - Critical to Warning  
*Visible in UI*: Yes  
*Learn More*: Health Rules, Policies

**POLICY_OPEN_CRITICAL**

**Description**: A critical health rule was violated.

*Category*: Policy Violations  
*UI Display name*: Health Rule Violation Started - Critical  
*Visible in UI*: Yes  
*Learn More*: Health Rules, Policies

**POLICY_OPEN_WARNING**

**Description**: A warning health rule was violated.

*Category*: Policy Violations  
*UI Display name*: Health Rule Violation Started - Warning  
*Visible in UI*: Yes  
*Learn More*: Health Rules, Policies

**POLICY_UPGRADED**

**Description**: A health rule violation was upgraded from warning to critical.

*Category*: Policy Violations  
*UI Display name*: Health Rule Violation Upgraded - Warning to Critical  
*Visible in UI*: Yes  
*Learn More*: Health Rules, Policies
RESOURCE_POOL_LIMIT

Description: A resource pool limit, such as a thread pool or connection pool, has been reached.

Category: Code Problems

UI Display name: Resource Pool Limit Reached

Visible in UI: Yes

Learn More: Remediation Actions, Policies

RUNBOOK_DIAGNOSTIC_SESSION_END

Description: A diagnostic session that was started by a diagnostic action triggered by a policy, has ended.

Visible in UI: No

Learn More: Diagnostic Sessions, Diagnostic Actions, Policies

RUNBOOK_DIAGNOSTIC_SESSION_FAILED

Description: A diagnostic session that was started by a diagnostic action triggered by a policy failure.

Visible in UI: No

Learn More: Diagnostic Sessions, Diagnostic Actions, Policies

RUNBOOK_DIAGNOSTIC_SESSION_STARTED

Description: A diagnostic session that was started by a diagnostic action triggered by a policy, session started.

Visible in UI: No

Learn More: Diagnostic Sessions, Diagnostic Actions, Policies

RUN_LOCAL_SCRIPT_ACTION_END

Description: A local script that was started by a remediation action triggered by a policy, has ended.

Visible in UI: No

Learn More: Remediation Actions, Policies

RUN_LOCAL_SCRIPT_ACTION_FAILED

Description: A local script that was started by a remediation action triggered by a policy, has failed.

Visible in UI: No

Learn More: Remediation Actions, Policies

RUN_LOCAL_SCRIPT_ACTION_STARTED

Description: A local script that was started by a remediation action triggered by a policy, has started.

Visible in UI: No

Learn More: Remediation Actions, Policies

SERVICE_ENDPOINT_DISCOVERED
A new service endpoint has been added to the application.

**Category:** Discovery

**UI Display name:** New Service Endpoint Discovered

Visible in UI: Yes

Learn More: Policies

---

A business transaction is now slow.

**Category:** Slow Transactions

**UI Display name:** Slow Transactions

Visible in UI: Yes

Learn More: Troubleshoot Slow Response Times, Transaction Thresholds

---

An SMS was sent to notify the recipient of an event.

Visible in UI: No

Learn More: Notification Actions

---

A business transaction has stalled.

**Category:** Slow Transactions

**UI Display name:** Transaction Stall

Visible in UI: Yes

Learn More: Troubleshoot Slow Response Times, Transaction Thresholds

---

Thrown when events occur during workflow execution.

**Category:** AppDynamics Data

**UI Display name:** Automation Event

Visible in UI: Yes

---

A thread dump action ended.

Visible in UI: No

Learn More: Diagnostic Actions, Policies

---

A thread dump action failed.

Visible in UI: No
Learn More: Diagnostic Actions, Policies

THREAD_DUMP_ACTION_STARTED
Description: A thread dump action started.
Visible in UI: No
Learn More: Diagnostic Actions, Policies

TIER_DISCOVERED
Description: A new tier has been added to the application.
Category: Discovery
UI Display name: New Tier Discovered
Visible in UI: Yes
Learn More: Policies

VERY_SLOW
Description: A business transaction is now very slow.
Category: Slow Transactions
UI Display name: Very Slow Transactions
Visible in UI: Yes
Learn More: Troubleshoot Slow Response Times, Transaction Thresholds

WARROOM_NOTE
Description: A War Room Note has been added.
Learn More: Virtual War Rooms
Monitor Infrastructure

While business transaction performance is the typical focus of a performance monitoring strategy, infrastructure performance can add insight into underlying factors in business transaction performance. Infrastructure Visibility provides visibility into the underlying OS infrastructure and networks on which your applications run. These agents enable you to isolate, identify, and troubleshoot infrastructure problems that can affect application performance such as resource-hogging tiers.

AppDynamics provides preconfigured application infrastructure metrics and default health rules to enable you to discover and correct infrastructure problems. You can also configure additional persistent metrics to implement a monitoring strategy specific to your business needs and application architecture.

In addition to health rules, you can view infrastructure metrics in the Metric Browser. Correlation Analysis and Scalability Analysis charts in the browser can be particularly useful for understanding how infrastructure metrics correlate to business transaction performance.

For more information, see Overview of Infrastructure Visibility.
Alert and Respond

About Alert and Respond in AppDynamics

AppDynamics can generate notifications or take other types of actions based on conditions or events you configure. Using the alert and respond feature, you can find out about problems as they happen, or even before they happen when you define alerts on warning conditions.

In AppDynamics, policies serve as the central configuration artifact for the alert and respond feature. A policy ties one or more conditions or events to the measures to take when the condition is met, or an event happens.

The condition or event is defined by a health rule, while an action encapsulates the steps to take. AppDynamics comes with several preconfigured health rules, giving you a head start and examples for you to follow when creating your own. For example, built-in health rules test for whether the Business Transaction error rate is much higher than normal or CLR Garbage Collection Time is too high. See Default Health Rules for more.

Actions automate the response to an event, such as the sending of an alert or performing diagnostic or remediation actions. See Alert and Respond API to learn how to create custom URLs for notifications.

While policies generate real-time responses to detected conditions, email digests generate email messages about the conditions and events in a system on a scheduled basis.

Notification actions that use email or SMS and email digests require that the SMTP server be configured for the Controller. See Enable an Email Server.

Permissions

Different users with different roles typically set up and use the various alert and respond features. Some permissions can be granted at the account level and some can be granted at the application or even the tier level.

Account Level Permissions

Email templates, HTTP request templates, and Email/SMS configuration are account-level features. The scope for these features is the entire AppDynamics account and all the applications in that account. Users who have account-level roles that include the necessary permissions can create and manage account-level templates and configuration. These permissions belong to the predefined Account Owner role. The Account Owner can also create custom roles that include some of these permissions. For example, an account owner can create an email template manager role and assign that role to users, who can then create and modify email templates.

To create, manage, and configure these account-level features, you need the following permissions respectively:

- Configure Email Templates
- Configure HTTP Request Templates
- Configure Email / SMS
Application and Tier-Level Permissions

Policies, health rules, actions, and email digests are application-level or tier-level features. The scope of these features is the application or tier in which they were created. To create, manage, and configure these features, you need the following permissions respectively:

- Configure Policies: Create, edit or delete policies
- Configure Health Rules: Create, edit, or delete health rules
- Configure Actions: Create, edit, or delete actions on agent properties UI
- Configure Actions: Create, edit, or delete email digests

Alert and Respond Policy Structure

As shown in the following diagram, policies match event triggers with the action to be taken in response to those triggers.

Alert and Respond Across the Platform

The alert and respond features work across AppDynamics products, including Infrastructure Visibility, Analytics, EUM, and Application Monitoring. Unless otherwise noted, this documentation describes the features in the context of Application Monitoring, which, by its nature, offers the broadest range of configuration and use case options. Specific features as described may not apply to other AppDynamics products.

Additional usage notes include:

- Policy triggers for applications can be health rule violation events or other types of events. Policy triggers for databases and analytics must be health rule violation events.
- The types of actions that you can create for an application include notifications, diagnostics, remediation, HTTP requests, custom actions and cloud auto-scaling. The types of actions that you can create for a database or analytics are limited to notifications, HTTP requests and custom actions.
- The types of entities affected by a health rule are more limited for databases and analytics than for applications.
- For information on using policies triggered by browser synthetic events, see ‘Alerting and Synthetics’ in Browser Synthetic Monitoring.
Example Use Cases

The following use cases illustrate some of the ways you can use alert and response features:

- Define health rules that apply to specific tiers or nodes. Instead of choosing specific nodes, you can trigger a rule when more than a certain percentage of nodes are unhealthy, say 20%.
- Start a diagnostic action for a business transaction.
- Alert when an app agent stops reporting to the Controller.

Create a node health rule based on the value of the Availability metric reported by the agent. If Availability is less than one, the agent is not reporting.

Alert when the 95th percentile metrics for specific business transactions reach a certain value. You want to apply this rule only to business transactions with names beginning with User.
You can generalize a health rule by specifying a relative metric path, rather than a specific metric. The health rule is evaluated for each of the affected business transactions. Use a relative metric path when you need to evaluate a single metric for multiple entities.

You have a large operation with several development teams, each responsible for a different service. You create a health rule for one service and then copy it. Then create different policies in which you can pair each copy of the health rule to an alert addressed to the appropriate team.

Start a script to change the size of the connection pool. You have an application that performs well for normal load. However, peak loads can cause the application to slow. During peak load, the AppDynamics not only detects the connection pool contention but also allows you to create a remediation script that can automate increasing or decreasing the size of the
connection pool. You can require human approval to run this script or just configure it to automatically execute when it is triggered. Create a runbook and associate it with a policy so that it will fire when the connection pool is exhausted.

You can specify any script or executable and the Java Machine Agent will execute it, and upload the results to the controller. You can download the script output on the Events screen (for events that trigger Policies).

- Alert when the available disk volume is low. Use an expression over two metrics—available and used disk space—to be alerted when disk volume is low.

---

**Edit Health Rule - CPU utilization is too high**

<table>
<thead>
<tr>
<th>Overview</th>
<th>Affected Entities</th>
<th>Critical Criteria</th>
<th>Warning Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Add Condition**
- **Copy from Warning Criteria**

If all of the following conditions are met:

- **Condition 1**
  - Metric Expression: $({{Avail}}/({{Avail}}+{{Used}}))*100
  - Evaluate to true on no data
  - Specific Value: 10
A policy consists of a trigger based on one or more events and an action in response to the trigger. You can use policies to automate monitoring, alerting, and problem remediation.

View Policies

To view and create policies, access the Policies UI by selecting Alert & Respond > Policies.

The policy list displays all the policies created for your application, with its triggers and actions taken. You can modify a policy by double-clicking it in the policy list.

Policy Triggers

Policy triggers are events that cause the policy to fire. The events can be health-rule violation events or other types of events, such as hitting a slow transaction threshold or surpassing a resource pool limit. See Health Rules, Troubleshoot Health Rule Violations and Monitor Events.

The triggering events can be broad, affecting any object in the application, or very narrow, affecting a specific object. For example, you can have a broadly-defined policy that fires whenever a resource pool limit—such as > 80% usage of EJB pools, connection pools and/or thread pools—is reached for any object in the application. An example of a narrowly defined policy is one that would fire only when existing health rules on memory utilization or JVM garbage collection time are violated.

A policy is triggered when at least one of the specified triggering events occurs on at least one of the specified objects.

Policy Actions

You can assign the actions that are taken when a policy is triggered, such as an email or SMS notification. Other types of actions do more than just notify. For example, a resource pool violation can trigger a script that increases the pool size.

Other common actions include restarting an application server if it crashes, purging a message queue that is blocked, or triggering the collection of transaction snapshots. You can also trigger a custom action to invoke third-party systems.

Because the definition of health rules is separate from the definition of actions, you can specify different actions for an event in different contexts, such as a threshold crossing in either a tier or node context.

See Build a Custom Action for information about custom actions. See Actions for more information about the different types of actions.

Policy Actions in Batch

You can configure a policy to execute its actions either:

- Immediately for every triggering event.
  For example, if in a two-second period a policy matched 100 events, it would start its actions 100 times as soon as each event occurred.
- Once a minute for all the events that triggered over the last minute. This is the batch option. The Execute actions in batch check box is selected by default.
  For example, if in a two-second period, a policy matched 100 events and then no triggering events occurred for the next 58
seconds, the policy would start each action just once. The context for the actions would be all 100 events.

Which you choose depends primarily on the type of action. For a notification action, it probably doesn’t make sense to send 100 emails or SMS messages in a few seconds. In this case, it makes sense to batch the actions with a summary of the events occurring during the last minute. This can be easily accomplished using an email template that iterates through the event list. See the example in Predefined Templating Variables.

However, if the actions are thread dumps, there is no reason to expect that all 100 events are on the same node. They might be on different nodes. For that kind of action, you would probably want the thread dump to be taken for each event and also, not to wait another 58 seconds before taking the thread dump.
Configure Policies

On this page:
- Permissions
- Policy Setup Wizard
- Create Policy Manually

Related pages:
- Policies
- Monitor Events
- Health Rules
- Troubleshoot Health Rule Violations
- Actions
- Enable an Email Server

There are two ways to configure policies:
- Use the Policy Setup Wizard for simple policies that send an email notification when a health rule is violated.
- Configure the policy manually for anything more complicated.

Permissions

To configure policies, you need the Configure Policies permission.

Policy Setup Wizard

For simple policies, in the Controller UI, click Alert & Respond > Policies > Policy Setup Wizard.

To create a notification policy, you must set up an SMTP server and supply an email address. The policy is then created.

To modify the policy later, select the policy and click the Edit button.

Create Policy Manually

To create a policy manually, in the Controller UI, click Alert & Respond > Policies > Create Policy Manually.

The Create Policy window contains the following panels:
- Trigger: Sets the events that trigger the policy, entities that are affected by the policy.
- Health Rule Scope: Defines the Health Rules that trigger the policy.
- Object Scope: Defines the objects that trigger the policy for application and user experience contexts.
- Actions: Sets the actions to take when the policy is triggered.

In either panel, you can configure the policy name, enabled status, and whether to batch the actions executed by the policy.

To create policies:

1. Click Alert & Respond in the menu bar.
2. Click Policies either in the right panel or the left navigation pane.
3. Select the context for the policy (specific Application, User Experience, Databases, Servers, or Analytics) from the pulldown menu.
4. To create a new policy, click the Create Policy (+) button.

You can edit, copy, enable or disable, and delete a policy by clicking the appropriate button.

Configure Policy Triggers

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

For policy triggers that depend on health violation events, you must create the health rules before you can create a policy that uses
To configure policy triggers:

1. Click **Create Policy** (+ button) or select a policy and click **Edit** (pencil button).
2. Enter a name for the policy in the **Name** field.
3. To enable the policy, check the **Enabled** check box. To disable the policy, clear the **Enabled** check box.
   You can also enable or disable a policy by selecting it in the policy list and clicking the Enable or Disable button.
4. If you check the **Execute actions in batch** check box, the policy fires its actions once for all the triggering events that occurred in the last minute. See ‘Policy Actions in Batch’ in **Policies**.
5. Select the **Trigger** panel if it is not already selected.
6. Check one or more types of events that should trigger the policy. You may need to click the arrow to expose specific events within an event category.
   If you check at least one health rule violation event, you can choose whether any or all health rule violations or only specific health rule violations will trigger the policy.

<table>
<thead>
<tr>
<th>Name</th>
<th>Enabled</th>
<th>Execute actions in batch</th>
</tr>
</thead>
</table>

**This Policy will fire when any of these Events occur**

- **Health Rule Violation Events**
  - Health Rule Violation Started - Warning
  - Health Rule Violation Started - Critical
  - Health Rule Violation Continues - Warning
  - Health Rule Violation Continues - Critical
  - Health Rule Violation Upgraded - Warning to Critical
  - Health Rule Violation Downgraded - Critical to Warning
  - Health Rule Violation Ended - Warning
  - Health Rule Violation Ended - Critical
  - Health Rule Violation Canceled - Warning
  - Health Rule Violation Canceled - Critical

- **Other Events**
  - Slow Transactions
  - Code Problems
  - Application Changes
  - Server Crashes
  - AppDynamics Config Warnings
  - Discovery
  - Synthetic Availability
  - Synthetic Performance
  - Mobile Crash
  - Errors

If your AppDynamics environment includes browser synthetic monitoring, you will see additional other events for Synthetic Availability and Synthetic Performance. See ‘Alerting and Synthetics’ in **Browser Synthetic Monitoring**.
You can optionally designate specific custom events to trigger the policy, using the Custom Events panel in the lower right corner. Click the + icon.
Configure Health Rule Violation Triggers

The Health Rule Scope panel determines the scope of the health rule violations that trigger the policy. If you have not created a health rule, you can create one by clicking the Create Health Rule button in this panel.

To configure the health rule violation triggers:

1. If you have not already done so, edit the policy to which you want to add Health Rule triggers.
2. Select the Health Rule Scope panel, then select Any Health Rules if you want the policy triggered by violation of any health rule.
3. To designate specific health rule violations to trigger the policy, click the + icon, and then choose the health rules from the embedded health rule browser. Click Create Health Rule to create a new health rule as the trigger for this policy.

Configure Object Scope

The Object Scope panel defines the objects the policy is triggered on. The object scope applies only to business transaction performance type health rules and node health type health rules which comprise tiers and nodes.

To configure objects to be monitored:

1. When you have finished selecting the health rule violations that trigger the policy, click the Object Scope panel to select objects to be monitored in case of an event. If you select Any Object, the policy will be triggered by the configured events when they occur on any object in your application.
2. To restrict the policy to specific objects, select **These Specified Objects** and then choose the objects. For example, the following policy fires when selected events occur on the ECommerce Server tier. You can similarly restrict the objects to specific nodes, business transactions, Ajax Requests, and so forth.

You can restrict the affected nodes on the node name, on the type of node (such as Java and .NET) on nodes in certain tiers, or on criteria such as meta-info, environment variables, and JVM system environment properties. Meta-info includes key-value pairs for:

- **key**: `supportsDevMode`
- **key**: `ProcessID`
- **key**: `appdynamics.ip.addresses`
- any key passed to the agent in the `appdynamics.agent.node.metainfo` system property

To trigger by Health Rule Violation Events, leave the selection at **Any Object**. Selecting **These specified objects** means that events occurring within the selected objects, for example, slow transactions, errors, and so forth, trigger the policy.

3. Click **Save**.

**Configure Policy Actions**

The policy actions panel binds an action to the trigger. It defines which actions the policy automatically initiates when the trigger causes the policy to fire.

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

**To configure policy actions:**

1. If you have not already done so, edit the policy to which you want to add actions. See **To Access the Policy Wizard**.
2. Select the **Actions** panel.
3. Click the + icon. The list of existing actions appears. The available actions vary depending on the product area selected for the policy, such as Applications, Servers, Databases and so on. You can filter the list by checking the check boxes for the types of actions you want to see.
4. In the list of actions, select the action that you want this policy to execute and click **Select**. If you do not see an appropriate action for your needs, click **Create**. For information on creating actions, see **Actions**. After you have created the action, select it here to assign it to the policy that you are configuring.
5. Click **Save**.

You can optionally test whether your action will be fired using the **Test Action Execution**. Ensure that the policy you are testing is enabled.
Test Action Execution

An event represents an error or exception generated by the application, the crossing of a performance threshold, or an operational change in the application, such as a JVM restart.

You can view and analyze events in the Controller UI. From the Database Visibility, Server Visibility, or user experience application pages in the UI, you can view events by clicking Events from the left navigation tree. The Events list also provides information on which policy actions are configured to fire when a specific event occurs. It does not execute the actual actions configured for the policy.

This information helps answer the following questions:

- Why didn’t my policy action execute? Perhaps policies are not actually configured to respond to an event as you thought they were.
- Why am I getting multiple actions, such as multiple emails, for the same event? Perhaps more than one policy is configured to perform the same action.

To see the actions that are triggered for an event:

1. In the events list, select the event for which you expect actions to fire.
2. From the Actions menu, select Test Action Execution.
Anomaly Detection

Related topics:
- Enabling and Configuring Anomaly Detection
- Troubleshooting Anomalies

AppDynamics Cognition Engine powers Anomaly Detection and Automated Root Cause Analysis, features designed to reduce Mean Time To Resolution for application performance problems. Anomaly Detection automatically tells you whether every Business Transaction in your application is performing normally. Automated Root Cause Analysis helps you quickly determine the root cause for problems revealed by Anomaly Detection.

Anomaly Detection and Automated Root Cause Analysis are available to SaaS customers only.

Anomaly Detection uses Machine Learning to reveal problems automatically

Machine Learning capabilities in Cognition Engine enable Anomaly Detection to estimate Expected Ranges for Business Transaction metrics. When Average Response Time (ART) or Errors Per Minute (EPM) deviate from their Expected Ranges in significant ways, Anomaly Detection alerts you that the Business Transaction has an Anomaly.

An Anomaly is a pattern of abnormal behavior of ART, EPM, or both, for one Business Transaction.

Every Anomaly is a mostly continuous series of Anomaly events. Every Anomaly event has a Severity Level of either Warning or Critical. Anomalies are displayed as timelines which highlight state change events—events where severity is upgraded (changing from Warning to Critical) or downgraded (changing from Critical to Warning).

Anomaly Detection and Health Rules complement each other

While both Anomaly Detection and Health Rules alert you to performance problems in your application, Anomaly Detection provides powerful insights that would be hard or impossible to obtain using Health Rules.

<table>
<thead>
<tr>
<th>Anomaly Detection</th>
<th>Health Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anomaly Detection uses Machine Learning to discover the normal ranges of key Business Transaction metrics, and alerts you when these metrics deviate significantly from expected values. This enables Anomaly Detection to identify a wider range of problems than a person could capture in Health Rules.</td>
<td>Health Rules are manually created to apply logical conditions that one or more metrics must satisfy. They are perfect for capturing the clear-cut logic of SLAs, for example.</td>
</tr>
<tr>
<td>Anomaly Detection requires no configuration except when you want to limit Anomaly alerting.</td>
<td>AppDynamics provides a default set of Health Rules and you create additional Health Rules manually as desired, configuring Time Periods, Trends, and schedules.</td>
</tr>
<tr>
<td>An Anomaly is a series of Anomaly events.</td>
<td>A Violation is a series of Health Rule violation events.</td>
</tr>
<tr>
<td>Anomalies are associated with Business Transactions.</td>
<td>Health Rules can apply to any entity.</td>
</tr>
</tbody>
</table>

Automated Root Cause Analysis saves time and effort

When a Business Transaction in your application has an Anomaly, you will want to know why. AI capabilities in Cognition Engine enable Automated Root Cause Analysis to monitor the health of all entities in your application, and show you Suspected Causes for every Anomaly. You can confirm or negate Suspected Causes with just a brief look, and drill down into deviating metrics and snapshots as desired. In this way, you quickly determine the root cause for application performance problems.
Enabling and Configuring Anomaly Detection

On this page:
- Enable Anomaly Detection
- Let Anomaly Detection Learn your Application
- Monitor Anomalies
- Configure Anomaly Detection (Optional)

Related topics:
- Anomaly Detection
- Troubleshooting Anomalies

Anomaly Detection must be enabled, but requires no configuration except when you want to limit anomaly alerting. Enabling Anomaly Detection also enables Automated Root Cause Analysis.

Anomaly Detection and Automated Root Cause Analysis are available to SaaS customers only.

Enable Anomaly Detection

Enable Anomaly Detection separately for each application.

- In Alert & Respond > Anomaly Detection, choose the desired application from the drop-down menu, and Toggle Anomaly Detection to ON

Let Anomaly Detection Learn your Application

After you enable Anomaly Detection, it takes 48 hours for Anomaly Detection and Automated Root Cause Analysis to become available. During that time, the Machine Learning models train on the Business Transactions in your application.

To view Business Transaction training status:

- In Alert & Respond > Anomaly Detection, view the Model Training tab
The table below enumerates the five possible training statuses of a Business Transaction.

<table>
<thead>
<tr>
<th>Status</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>In Training</td>
<td>Model training is in progress for the Business Transaction</td>
</tr>
<tr>
<td>Ready</td>
<td>Model training is complete, and the Business Transaction is healthy</td>
</tr>
<tr>
<td>Warning</td>
<td>Model training is complete, but the Business Transaction has experienced one or more Warning level anomalies during the training period</td>
</tr>
<tr>
<td>Critical</td>
<td>Model training is complete, but the Business Transaction has experienced one or more Critical level anomalies during the training period</td>
</tr>
<tr>
<td>Not Available</td>
<td>Model training is incomplete, and the Business Transaction is not visible to Anomaly Detection</td>
</tr>
</tbody>
</table>

The models continue training as long as Anomaly Detection is enabled. If traffic to a Business Transaction is interrupted for long enough to prevent training that day, Anomaly Detection continues to function using the models from the previous 7 days.

Monitor Anomalies

View the Applications > Business Transactions screen, and for any Business Transaction of interest, click the Warning or Critical icon in the Health column:

A list of Health Rule Violations and Anomalies for that Business Transaction appears:
Since there are multiple ways to view lists of anomalies, monitoring anomalies can be a natural extension of the way you work with AppDynamics.

Choose any of the following options to open a detailed view that includes the results of Automated Root Cause Analysis.

If you set up and validate tools for a tools team

- In Alert & Respond > Anomaly Detection, view the Anomalies tab

If you monitor applications for an application ops team

- In Applications > Events, filter Event Types to include Anomalies
1. In Applications > Troubleshoot > Violations & Anomalies, filter Event Types to include Anomalies.

You can configure policies to be triggered by anomalies, similarly to the way you configure policies to be triggered by Health Rules.

**Configure Anomaly Detection (Optional)**

By default, Anomaly Detection alerts you about all Anomalies for all Business Transactions in your application.

You can configure Anomaly Detection to only surface Anomalies for the combination of Business Transactions and severity levels that you specify. Do this if you prefer to see fewer, more narrowly focused alerts.

Automated Root Cause Analysis does not need to be enabled or configured.

To configure Anomaly Detection:

1. Click **Configure Anomaly Detection** to open the configuration dialog.

2. If desired, limit the Business Transactions on which you want Anomaly Detection to alert
   - Default (all Business Transactions) is shown.
3. If desired, limit alerts to either **Warning** or **Critical** severity only
   - Default (All Severities, meaning both Warning and Critical) is shown
Troubleshooting Anomalies

On this page:

- Drill into an Anomaly
  - Examine the anomaly description
  - Examine the timeline
  - Examine the flow map
  - Examine the Top Suspected Causes
- Drill into a Suspected Cause
  - Examine Top Deviating Metrics for the Business Transaction
  - Examine Suspected Cause Metrics
- Takeaways
- (Optional) Inspect Snapshots and Execute Actions

Related topics:

- Anomaly Detection
- Enabling and Configuring Anomaly Detection

To demonstrate techniques for using Anomaly Detection and Automated Root Cause Analysis effectively, this example follows an anomaly from the moment it surfaces until its root cause is confirmed. The troubleshooting process can begin with any of the multiple ways to view anomalies. Let's assume that you begin with the Alert & Respond > Anomaly Detection screen.

Anomaly Detection and Automated Root Cause Analysis are available to SaaS customers only.

Drill into an Anomaly

- In Alert & Respond > Anomaly Detection, view the Anomalies tab

- Doubleclick an anomaly to open the detailed view
Initially, everything you see describes what is happening at the anomaly’s Start Time. To see how things change later in the anomaly’s lifecycle, click events further along the timeline.

**Examine the anomaly description**

The anomaly description tells you what the anomaly is about by naming the Business Transaction that has the anomaly, the severity level of the selected state transition event, and the top deviating Business Transaction metrics.

For our example, these are:

- **Business Transaction**: `/r/Checkout`
- **Severity Level**: **Critical**
- **Top Deviating Metrics**: **Average Response Time**

In this case, since the deviating metric is Average Response Time, it’s clear that the problem is Checkout responding slowly.

**Examine the timeline**

Note the state transition events, which mark the moments when the anomaly moves between Warning and Critical states.

- The timeline in this example begins in the **Critical** state, followed 30 minutes later by a transition to the Warning state, which lasts only 8 minutes
- Since this fairly simple anomaly starts in the Critical state and remains there for most of its lifecycle, we can probably learn all we need to know from the initial event

By contrast, patterns that appear in more complicated timelines may help you understand anomalies. For example, this timeline from a different anomaly repeatedly toggles from a brief Warning state to a longer Critical state:
In such a case, it would be wise to examine several state change events, to see what clues the toggling between states offers about problems in your application.

**Examine the flow map**

In our example flow map:

- The START label shows that the Business Transaction begins with the **OrderService** tier
- Between **OrderService** tier and its numerous dependencies, two tiers are red—these are the tiers where the system has found Suspected Causes

Now we can focus on determining which of the red tiers harbors the root cause of the anomaly.

**Anomaly Detection flow maps are different**

There are two kinds of flow maps in AppDynamics: (1) the Anomaly Detection and Automated RCA flow map described on this page, and (2) the Business Transaction flow map. Each of these detects deviating or unhealthy entities in its own way. Therefore, you will see some differences:

- The two flow maps may show a different health status (as represented by color) for the same entity, because each one uses its own algorithm to determine health
- User preferences for positioning or hiding entities saved for the Business Transaction flow map have no effect on the Anomaly Detection flow map
- Some links between tiers might be shown in one type of flow map but hidden in the other
  - For example, when no data is flowing through a tier or between tiers:
    - the Business Transaction flow map may hide them, as ‘inactive’ tiers or links
    - the Anomaly Detection flow map may show them in order to represent the application topology completely
Examine the Top Suspected Causes

The Top Suspected Causes show likely root causes of a Business Transaction performance problem. In this case, we want to know why Checkout is responding slowly.

The first Suspected Cause is a Process CPU Burnt issue on the `eretail.prod.payment01_1` node of the PaymentService1 tier:

Hover over the Suspected Cause to highlight the relevant entities in the flow map. Everything but the critical path fades away, revealing that `OrderService`, where the Business Transaction starts, and which had a degraded response time, relies on `PaymentService1`:

The second Suspected Cause is an HTTP call on `OrderService` itself.

Hover to highlight the affected entities:
Which Suspected Cause is the root cause, and which is just a symptom of the overall problem?

- We have a plausible root cause in the Process CPU Burn issue on PaymentService1 tier, which is ranked likeliest by the system.
- Meanwhile, the HTTP call on OrderService bears some analysis:
  - An HTTP call includes both a request and a response
  - We know that the tier on other end, PaymentService1, has its own problem
  - Therefore, we can infer that the HTTP response from PaymentService1 is what makes the call slow

Now we see that both Suspected Causes originate with PaymentService1, and the HTTP call issue is really a side-effect of the Process CPU Burn issue. The system's ranking makes sense.

As we continue to investigate, if we decide that the Process CPU Burn issue is not the root cause after all, we can reconsider the HTTP call.

There can be anywhere from zero to three Top Suspected Causes. For example, if ART is high but every entity connected with ART is behaving normally, there are zero suspected causes because no suspected cause can be identified.

Drill into a Suspected Cause

Click More Details for the Suspected Cause to see (1) a simplified timeline, and (2) metrics graphed over time. Two kinds of graphed metrics appear: Top Deviating Metrics for the Business Transaction, and Suspected Cause Metrics.

**Examine Top Deviating Metrics for the Business Transaction**

Deviating Business Transaction metrics can indicate why an anomaly was important enough to surface. (The system does not surface anomalies for every transitory or slight deviation in metrics. Such anomalies would be of dubious value, since their customer impact is minimal. For the same reason, anomalies are surfaced for Business Transactions which have a CPM of under 20.)

Each deviating metric is shown as a thin blue line (the metric's value) against a wide gray band (the metric's Expected Range). You can:
- Scroll along the graph to compare a metric's value with its Expected Range at any time point
- Hover over a time point to view the metric's value and Expected Range in numerical form

Hovering to view values in numerical form is essential when the Expected Range is so much smaller than the deviating metric values that the gray band “disappears” into the X axis.

In our example:

- The deviating metric spiked, remained elevated for about 30 minutes, then subsided back into Expected Range
- Seven minutes after the metric returned to its Expected Range, the Severity Level changed from Critical to Warning, and eight minutes after that, to Normal

Hovering over time points tells us that for the period of deviation:

- Average Response Time was around 1200 ms and above, while its Expected Range was from 370.08 to 721.24 ms

With a key metric this elevated, it made sense for the system to surface this anomaly.

Because Top Deviating Metrics are at Business Transaction level, they are the same regardless for all Suspected Causes.

**Examine Suspected Cause Metrics**

You view, scroll through, and hover over Suspected Cause Metrics in the same way that you do Top Deviating Metrics.

Top Deviating Metrics describe the Business Transaction, while Suspected Cause Metrics describe tiers or nodes further down in the entity tree. This means that if you have ART as a Top Deviating Metric and as a Suspected Cause Metric, *those are two different metrics.* Likewise, EPM as a Top Deviating Metric and as a Suspected Cause Metric are also two different metrics.

While the Suspected Cause Metric likely contributes to the way the Top Deviating Metric behaves, the values of the two metrics will differ.

In our example,

- Suspected Cause Metrics are shown for the `eretail.prod.payment01_1` node within the `ProcessPayment1` tier
- That is the *only* node the tier has—if the tier had multiple nodes, metrics could be viewed separately for each node
- The pattern of elevation in the Process CPU Burnt and Process CPU Used metrics perfectly matches the pattern we saw in the Business Transaction metrics
The hypothesis we have been developing is now confirmed:

1. CPU usage spiked on ProcessPayment1, a tier that is downstream from the tier where the Business Transaction starts 
2. This slowed down response time on ProcessPayment1, including its HTTP response to the HTTP request from OrderService 
3. The slow HTTP call in turn slowed response time on OrderService 
4. Since OrderService is where the Checkout Business Transaction starts, Checkout has an slow response time anomaly 
5. Since the Process CPU Burn issue on ProcessPayment1 is the Suspected Cause that's deepest in the entity tree, that is the root cause of the anomaly 

Takeaways

We used Anomaly Detection and Automated Root Cause Analysis to quickly find the root cause of a Business Transaction performance problem. What kind of time and effort did Anomaly Detection and Automated Root Cause Analysis save us in this case?

Recall that the tier where the Business Transaction started, OrderService, has multiple dependencies including other services and datastores. Anomaly Detection and Automated Root Cause Analysis eliminated (1) all but two tiers as origins of the slow response time on OrderService, and (2) all but the most relevant of the many metrics on those tiers.

You were spared the tedious process of investigating multiple metrics on each dependency in turn. Instead, you confirmed or negated Suspected Causes with a quick glance at timelines, flow maps, and metrics. Anomaly Detection and Automated Root Cause Analysis performed the vast majority of the work in root cause analysis, presenting you with the information you needed to quickly form and verify a hypothesis.

(Optional) Inspect Snapshots and Execute Actions

When you view an anomaly, you can inspect

- Business Transaction snapshots from the time period of the anomaly
- Actions executed as the result of policies you have configured for the Business Transaction

These are options, not required parts of the standard troubleshooting flow. They are typically done as follow-up.

In our example:

- Suppose we want more context for the Process CPU Burnt issue on PaymentService1. We can view snapshots of transactions affected by that issue. Doubleclick a snapshot in the list to open it in its own window, and if desired, drill down into details and call graphs.
It is common to send messages to a ticketing system when an anomaly occurs. In this case we posted to Slack, for our Ops team to see that on their phones.
Health Rules

On this page:

- How to Set Up Health Rules?

Related pages:

- Configure Health Rules
- Troubleshoot Health Rule Violations
- Policies
- Actions

This topic introduces health rules, the policy statements that define triggers in AppDynamics policies.

What is a Health Rule?

Health rules let you specify the parameters that represent what you consider normal or expected operations for your environment. The parameters rely on metric values, for example, the average response time for a business transaction or CPU utilization for a node.

When the performance of an entity affected by the health rule violates the rule’s conditions, a health rule violation occurs. The health statuses are represented as critical, warning, normal, and unknown.

When the health status of an entity changes, a health rule violation event occurs. Examples of health rule violation events are, a health rule violation

- starting
- ending
- upgrading from warning to critical or
- downgrading from critical to warning

The health statuses of entities and health rule violations are surfaced in the controller user interface. A health rule violation event can also be used to trigger a policy, which can initiate automatic actions, such as, sending alerting emails or running remedial scripts.

You create health rules using the health rule wizard, described in Configure Health Rules. The wizard groups commonly-used system entities and related metrics to simplify setting up health rules. You can also use the default health rules provided by AppDynamics, as-is or modify them.

Default Health Rules

AppDynamics provides a default set of health rules for some products, such as, applications and servers. These default health rules vary depending on the entity. To see the default rules, before any health rules have been added to your AppDynamics installation:

1. Select the Alert & Respond tab from the top navigation bar.
2. Click Health Rules in the left panel.
3. From the drop-down list in the right panel select the entity.
   The default health rules for the selected entity are displayed.

If any of these predefined health rules are violated, the affected entities are marked in the UI as yellow-orange if it is a Warning violation and red if it is a Critical violation.

In many cases, the default health rules may be the only health rules that you need. You can edit and customize the health rules to suit your application. You can also disable the default health rules.

Health Rule Scopes

The health rule scope determines the set of default health rule types. You can choose the scope to get a set of default health rule types for applications, servers, or databases. For example, when you define a mobile application as the scope, the default health rules such as, crash rates and HTTP/network error rates are displayed. Similarly, if you define the health rule scope for an application, the health rules would be for business transactions, CPU/memory utilization, so on.

From Alert & Respond > Health Rules, you can select one of the following health rule scopes from the drop-down list:

- Applications
- User Experience: Browser Apps
- User Experience: Mobile Apps
- Databases
- Servers
- Analytics
You can also create new health rules to add to the default set for each scope. You may want to add the health rule *app starts* to your mobile application. This health rule is not part of the default set of health rules in the mobile app scope, so you would just need to add a new health rule.

**Health Rule Types**

The health rule wizard groups health rules into types that are categorized by the entity that the health rule covers. This allows the wizard to display appropriate configuration items during the health rule creation.

The health rule types are:

- **Transaction Performance**
  - Overall Application Performance: Groups metrics related to load, response time, slow calls, stalls, with applications.
  - Business Transaction Performance: Groups metrics related to load, response time, slow calls, stalls, so on, with business transactions.

- **Node Health**
  - Node Health-Hardware, JVM, CLR: Groups metrics like CPU and heap usage, disk I/O, so on, with nodes.
  - Node Health-Transaction Performance: Groups metric related to load, response time, slow calls, stalls, so on, with nodes.
  - Node Health-JMX: Java only, groups metrics related to connection pools, thread pools, so on, with specific JMX instances and objects in specific nodes and tiers.

- **User Experience-Browser Apps**
  - Pages: Groups metrics like DOM building time, JavaScript errors, so on, with the performance of application pages for the end user.
  - iFrames: Groups metrics like first-byte time, requests per minute, so on, with the performance of iFrames for the end user.
  - AJAX Requests: Groups metrics like Ajax callback execution time, errors per minute, so on, with the performance of Ajax requests for the end user.
  - Virtual Pages: Groups metrics like End User Response Time, Digest Cycles, HTML Download Time, DOM Building Time, etc. for virtual pages created with Angular. See AngularJS Support for information on what these metrics mean in the context of virtual pages.

- **User Experience-Mobile Apps**
  - Mobile Apps: Groups metrics related to mobile app crashes, starts, and server calls as well as network requests and errors.
  - Network Requests: Groups metrics like HTTP and network errors, request time, and requests per minute with network requests.

- **Servers**
  - Groups metrics related to hardware resources.

- **Databases & Remote Services**
  - Groups metrics related to response time, load, or errors with databases and other backends.

- **Advanced Network**
  - Groups metrics related to Network Visibility, such as PIE (performance impact events), zero window, data retransmission, and errors.

- **Error Rates**
  - Groups metrics related to exceptions, return codes, and other errors with applications or tiers.

- **Information Points**
  - Groups metrics like response time, load, or errors with information points.

- **Service Endpoints**
  - Java and .NET only; groups metrics like average response time, calls per minute, and errors per minute with service endpoints.

- **Custom**
  - Presents all the metrics collected by the agent that could affect a single business transaction, a single node or overall application performance. Use this type to create rules that evaluate custom metrics.

When you select one of these health rule types, the wizard offers you the metrics commonly associated with that type in an embedded browser.

**How to Set Up Health Rules?**

AppDynamics recommends the following process to set up health rules for your application:

1. Identify the key metrics (performance indicators) on the key entities that you need to monitor.
2. Click **Alert & Respond > Health Rules** to examine any default health rules that are provided by AppDynamics.
   - Compare your list of metrics with the metrics configured for the default rules.
   - You can view the list of affected entities for each of the default health rules and modify them. See, Configure Affected Entities.
   - If the default health rules cover all the key metrics you need, determine if the pre-configured conditions are applicable to your environment. If required, modify the conditions.

Define a **metric expression** to evaluate complex criteria for a condition.

Define a **boolean expression** to evaluate multiple health rule conditions.
3. If default health rules do not cover all your requirements or if you need finely-applied health rules to cover specific use cases, create new health rules.
   a. Identify the type of health rule that you want to create. See Health Rule Types.
   b. Decide which entities are affected by the new rule. See Entities Affected by a Health Rule.
   c. Define the conditions to monitor. See Create and Configure Conditions.

4. If you want the health rules to be evaluated according to a pre-defined time schedule, create a health rule schedule. In some situations, a health rule is more useful if it is evaluated at a particular time. See Health Rule Schedules.

After you set up health rules you must configure policies and actions to be executed when health rules are violated. See Policies and Actions.

Additional Considerations

- Your application status is based on health rules for the current time range. If you disable old health rule policies, or enable new ones, you might see errors in red in your application status, even if there are no current critical events based on the new policies. To verify that your new or disabled health rule policies have taken effect, change the time range in your dashboard to a smaller, more recent time frame.

- When you are configuring health rules for business transactions with baseline selected in the configured condition with a very fast average response time (ART) such as 25 ms, using standard deviation as a criterion can cause the health rule to be violated too frequently. The health rule may violate too frequently because a tiny increase in response time can represent multiple standard deviations. In this case, consider adding a second condition that sets a minimum ART as a threshold. For example, if you do not want to be notified unless ART is over 50 ms, you could set your threshold as ART > 2 Standard Deviations and ART > 50 ms.

- Similarly, when configuring health rules for calls-per-minute (CPM) metrics with baseline selected in the configured condition, the health rule may never be violated if the condition is using standard deviations, and the resulting value is below zero. In this case, consider adding a second condition that checks for a zero value, such as CPM < 2 Standard Deviations and CPM < 1.
Health Rule Schedules

On this page:

- Health Rule Enabled Schedule
- Health Rule Evaluation Window
- Health Rule Wait Time After Violation

Related pages:

- Health Rules
- Configure Health Rules
- Troubleshoot Health Rule Violations
- Policies
- Actions

The metrics associated with a health rule are evaluated according to a schedule that you control. You can configure:

- when a health rule is in effect
- which data set should be used, based on time
- what special rules should be in place during a violation event

Time evaluation for health rule schedules is based on the time zone of the Controller, regardless of where an app agent is situated. For example, if a Controller is in San Francisco but the app agent is in Dubai, Pacific Time applies to the health rule schedule.

All SaaS Controllers use Pacific Time (PT).

Health Rule Enabled Schedule

By default, health rules are always enabled. You can define schedules for evaluation of the health rules. Built-in schedules exist for:

- End of business hours
- Weekday lunch
- Weekday mornings
- Weekdays
- Weekends

You can also configure your own schedules based on UNIX cron expressions using custom values.

Health Rule Evaluation Window

The health rule evaluation window is the period of time over which the data used to evaluate the health rule is collected.

Different kinds of metrics provide better results using different sets of data. You can manage how much data AppDynamics uses when it evaluates a particular health rule by setting the data collection time period. The default value is 30 minutes.

- For metrics based on an average calculation, such as, average response time, AppDynamics averages the response time over the evaluation window. A five-minute window means that the last five minutes of data is used to evaluate if the health rule was violated.
- For metrics based on a sum calculation, such as, the number of calls, AppDynamics uses the total number of calls counted during the evaluation window.

Health Rule Wait Time After Violation

The health rule wait time setting lets you control how often an event is generated while the conditions found to violate a health rule continue. If the controller determines that a health rule has been violated, with a status of either Critical or Warning, an Open Critical or Open Warning event is generated. This event is used to trigger any policies that match the health rule, and initiate any actions that the policies require.

Once an Open event has occurred, the controller continues to evaluate the status of the health rule every minute. If the controller continues to detect the same violation, the violation remains open with the same status. A corresponding Continues Critical or Continues Warning event may be generated to link to any related policy.
A Continues event every minute might be too noisy for your health rule. The health rule's Wait Time after Violation setting is used to throttle how often these Continues events are generated for continuing health rule violations. The default is every 30 minutes.

To use Continues Critical and Continues Warning events, adjust the default Wait Time after Violation value to the desired frequency. Then configure a policy matching the health rule with the Health Rule Violation Continues - Warning and/or Health Rule Violation Continues - Critical events selected in the Health Rule Violation Events section of the policy settings.

The violations displayed in the Health Rules Violations page, under Troubleshoot, are updated only when a health rule violation event is triggered.

If the Controller is unable to evaluate the rule—for example, if a node stops reporting—the Evaluation Status of the health rule is marked as a grey question mark or Unknown in the Current Evaluation Status tab in the right panel of the health rules list. The current violation event remains open until the Wait Time after Violation period has elapsed, at which point the violation event is closed and a new event is triggered, causing the Health status itself of the rule to display as Unknown.
Health Rule Entities

A health rule can evaluate metrics associated with an entire application or a limited set of entities. For example, you can create business transaction performance health rules that evaluate certain metrics for all Business Transactions in the application or node health rules that cover all the nodes in the application or all the nodes in specified tiers. The default health rules are in this category.

You can also create health rules that are applied to a limited set of entities in the application, or even a single entity such as a node or a JMX object or an error. For example, you can create a JMX health rule that evaluates the initial pool size and number of active connections for specific connection pools in nodes that share certain system properties.

Monitoring Serverless Entities

Serverless functions are tracked at the tier level. A serverless function is indicated by a lambda () icon inside each tier. When you configure a health rule for an application comprising serverless entities, you can choose to monitor the serverless tiers in the Affected Entities tab. For information on how various health rules are evaluated for serverless entities comprising tiers for AWS Lambda, see Evaluating Serverless Tiers.

The health rule wizard lets you specify which entities the health rule affects, enabling the creation of very specific health rules. For example, for a Business Transaction, you can limit the tiers that the health rule applies to, or limit the health rule application to specific business transactions by name or by names that match certain criteria.

For node health rules, you can specify the type of the node, such as Java, .NET, PHP, and so on.

You can specify that a health rule applies only to nodes that meet certain criteria.
Entities Affected by a Health Rule

For an **Overall Application Performance Health Rule** type, the health rule applies to the entire application, regardless of the business transaction, tier, or node.

If you configure your Health Rule to work with tiers, you must also configure the parallel policy to work with tiers. However, if you configure your Health Rule to work with tiers, but your policy is configured with nodes first, you will not trigger any actions or notifications. The inverse is also true. The following screenshots show examples of a health rule and a policy created in the correct order.
The following table lists the entities that you can apply health rules to.

<table>
<thead>
<tr>
<th>Health rule type</th>
<th>Applicable Entities</th>
</tr>
</thead>
</table>
| **Business Transaction Performance** | • All Business Transactions in the application  
• All Business Transactions within tiers that you select  
• All Business Transactions within serverless tiers that you select for a serverless platform*  
• Individual Business Transactions that you select  
• Business Transactions with names that have patterns matching criteria that you specify (such as all Business Transactions with names that start with "INV") |
| **Custom**                    | • A business transaction that you specify  
• A node that you specify  
• Overall application performance |
| **Databases & Remote Services health** | • All databases and remote services in the application  
• Individual databases and remote services that you specify  
• Databases and remote services with name matching criteria that you specify |
| **Error Rates**               | • All Errors in the application  
• Specific error types that you select  
• Errors with the specified tiers  
• Errors within the specified serverless tiers for serverless platform*  
• Errors with names that have patterns matching criteria that you specify |
| **Information Points**        | • All servers in the application  
• Information points that you specify  
• Information points with names matching criteria that you specify |
### Node Health—JMX
- All JMX instance names (MBeans) in the application
- Specific MBeans
- MBeans on certain nodes
- Specific JMX objects
- All nodes in the application
- Nodes within specified tiers
- Individual nodes that you specify
- Nodes with names, meta-data, environment variables or JVM system environment properties with matching criteria that you specify
- Nodes with certain MBeans

### Node Health—Transaction Performance or Node Health—Hardware, JVM, CLR
- All tiers in the application
- Individual tiers that you specify
- All serverless tiers in the application for a serverless platform*
- Individual tiers that you specify for a serverless platform*
- All nodes in the application
- Nodes types, such as Java nodes, PHP nodes, and so on
- Nodes within specified tiers
- Individual nodes that you specify
- Nodes with names, meta-data, environment variables or JVM system environment properties with matching criteria that you specify

*The performance of serverless tiers is not evaluated for Tier/Node Health (Hardware) health rules. AWS does not offer node-level dashboards or metrics because the serverless platform runtime instances spin up and down on demand.

### Server health
- All servers in the application
- Servers that you specify
- Servers in specific tiers

### Service Endpoint
- All service endpoints in the application
- Service endpoints that you specify
- Service endpoints with names matching criteria that you specify
- Service endpoints within specified tiers
- Service endpoints within specified serverless tiers for serverless platform*

*The performance of serverless tiers is not evaluated for Tier/Node Health (Hardware) health rules. AWS does not offer node-level dashboards or metrics because the serverless platform runtime instances spin up and down on demand.

### User Experience - Mobile Apps
- All mobile apps with the specified app key
- The specified mobile apps
- Mobile apps matching the given criteria

### User Experience - Mobile Network Requests
- All network requests in the application
- Network requests that you specify
- Network requests with names matching criteria that you specify
- Network requests of the specified mobile apps

### User Experience - Browser Apps—Pages, Iframes, Ajax Requests, Virtual Pages, Synthetic jobs
- All such entities
- Entities that you specify
- Entities with names matching criteria that you specify
Health Rule Conditions

You define the acceptable range for a metric by establishing health rule conditions. A health rule condition sets the metric levels that constitute a Warning status or a Critical status.

A condition consists of a Boolean statement that compares the current value of a metric against one or more static or dynamic thresholds based on a selected baseline. If the condition is true, the health rule violates. The rules for evaluating a condition using multiple thresholds depend on configuration.

Static thresholds are straightforward. For example, is a business transaction’s average response time greater than 200 ms? The condition is evaluated to ‘true’ if the average response time is greater than 200 ms and the health rule violates.

Dynamic thresholds are based on a percentage in relation to, or a standard deviation from, a baseline built on a rolled-up baseline trend pattern. A daily trend baseline rolls up values for a particular hour of the day during the last thirty days, whereas a weekly trend baseline rolls up values for a particular hour of the day, for a particular day of the week, for the last 90 days. For more information about baselines, see Dynamic Baselines.

You can define a threshold for a health rule based on a single metric value or on a mathematical expression built from multiple metric values.

The following are typical health rule conditions:

- If the value of the Average Response Time is greater than the default baseline by 3 X the Baseline Standard Deviation . . .
- If the count of the Errors Per Minute is greater than 1000 . . .
- If the number of MB of Free Memory is less than 2 X the Default Baseline . . .
- If the value of Errors per Minute/Calls per Minute over the last 15 days > 0.2 . . .
  This example combines two metrics in a single condition. You can use the expression builder embedded in the health rules wizard to create conditions based on a complex expression comprising multiple interdependent metrics.
- If the (average response time > baseline OR errors per minute > baseline) AND (calls per minute > the defined threshold) . . .
  This example uses multiple conditions to evaluate the health rules. You can use the 'CUSTOM' option to define a boolean expression to evaluate the conditions.

Critical and Warning Conditions

Conditions are classified as either critical or warning conditions.

Critical conditions are evaluated before warning conditions. If you have defined a critical condition and a warning condition in the same health rule, the warning condition is evaluated only if the critical condition is not true.
The configuration procedures for critical and warning conditions are identical, but you configure these two types of conditions in separate panels. You can copy a critical condition configuration to a warning configuration and vice-versa and then adjust the metrics in the copy to differentiate them. For example, in the Critical Condition panel you can create a critical condition based on the rule:

- If the Average Response Time is greater than 1000

Then from the Warning Condition panel, copy that condition and edit it to be:

- If the Average Response Time is greater than 500

As performance changes, a health rule violation can be upgraded from warning to critical if performance deteriorates to the higher threshold or downgraded from critical to warning if performance improves to the warning threshold.

**Evaluation Criteria**

When you define multiple conditions for a health rule, they are evaluated based on the criteria you define. You can use the following options to define the evaluation criteria:

- All - the health rule violates if all the conditions defined in the criteria evaluate to 'true'
- Any - the health rule violates if one of the conditions defined in the criteria evaluates to 'true'
- Custom - the health rule violates if the boolean expression with multiple conditions evaluates to 'true'

For information on how to configure evaluation criteria, see Configure Health Rule Evaluation Criteria.

The following table uses examples to illustrate how a health rule is evaluated based on the criteria and when is it considered to violate.
<table>
<thead>
<tr>
<th>Health rule configuration</th>
<th>Evaluation</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single condition</td>
<td>the condition evaluates to 'true'</td>
<td>A health rule that compares 'average response time' with a defined baseline.</td>
</tr>
</tbody>
</table>
| Multiple conditions with 'ANY' evaluation criteria | one of the health rule conditions evaluates to 'true' | A health rule that monitors the health of business transaction may measure any of the following performance metrics:  
  - average response time or  
  - errors per min |
| Multiple conditions with 'ALL' evaluation criteria | all of the health rule conditions evaluate to 'true' | A health rule that monitors the health of business transaction measures all of the following metrics:  
  - response time  
  - average response time greater than a baseline value, correlated with the application load  
  For example, 50 concurrent users on the system. A policy is defined such that a remedial action is initiated only if the load (calls per minute) is high although the response time threshold is reached. The first part of the condition evaluates the response time and the second part ensures that the health rule is violated only when there is sufficient load. |
| Multiple conditions with 'CUSTOM' evaluation criteria | the boolean expression with multiple conditions evaluates to 'true' | A health rule that monitors the health of a Business Transaction, measures the performance based on the following conditions:  
  - (average response time greater than baseline OR errors per min greater than baseline)  
  AND  
  - (calls per min greater than threshold) |

**Persistence Thresholds**

Temporary spikes in metric performance data is a major cause of false alerts. Persistence thresholds allow you to define a 'sensitivity level' for a health rule and thereby reduce the number of false alerts. You can define the 'number of times metric performance data should exceed the defined threshold' to constitute a violation and subsequently trigger an alert.

For example, when monitoring the CPU utilization, you would not want to be reported of a single violation (section A in figure) of the threshold. However, if the violation of threshold continues to occur multiple times (section B in figure) during the evaluation time period, you would want to be alerted.

![CPU Utilization Graph](image)
How are Conditions Evaluated if No Data is Reported?

The Evaluate to true on no data option controls the evaluation of the condition in cases where any metric on which the condition is based, does not return any data. The condition evaluates to 'unknown' (default) when no data is returned. If the health rule is based on all the conditions evaluating to true, having no data returned may affect whether the health rule triggers an action.

When you define a health rule evaluation time frame, reference data is collected for each data point. If the configured metric fails to report data during the time frame, the health rule condition is evaluated as follows:

<table>
<thead>
<tr>
<th>Evaluate to true on no data</th>
<th>Trigger only when violation occurs x times in the last y min(s)</th>
<th>Condition Evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enabled</td>
<td>Enabled</td>
<td>The condition is evaluated for each data point in the evaluation time frame. The condition evaluates to 'true' when metric fails to report any data for a given data point. For example, when you set the persistence threshold, X = 3 for an evaluation time frame, Y = 5. This means that 5 data points are required to evaluate the condition. Data is reported for 4 data points, no data is reported for 1 data point and the metric exceeds the threshold twice. The condition evaluates to 'true' for the minute when no data is reported.</td>
</tr>
<tr>
<td>Enabled</td>
<td>Disabled</td>
<td>The condition does not evaluate to 'true' if the configured metric fails to report data for any data point during the evaluation time frame.</td>
</tr>
<tr>
<td>Disabled</td>
<td>Disabled</td>
<td>The condition does not evaluate to 'true' if the configured metric fails to report data for any data point during the evaluation time frame.</td>
</tr>
</tbody>
</table>

Custom Boolean Expression

A condition consists of single or multiple statements that evaluate different metrics. You can define a single condition or multiple conditions to evaluate performance metrics of your application. When you define multiple conditions, you may want to define an evaluation criteria using a boolean expression.

Advantages of using a boolean expression are:

- eliminates the need to create multiple health rules to monitor various performance metrics. Using a boolean expression allows you to evaluate complex criteria for multiple conditions in one go.
- well-calibrated boolean expression ensures reduced false alerts.
- easy to create and maintain health rules with complex evaluation criteria using simple condition names. Conditions are named as A, B, C and so on.
- allows the use of AND and OR operators to define a highly complex boolean expression. You can use a maximum of 8 operators in your boolean expression.

Evaluation Scope

The health rule evaluation scope defines how many nodes in the affected entities must violate the condition before the health rule is considered violated.

Evaluation scope applies only to business transaction performance type health rules and node health type health rules in which the affected entities are defined at the tier level.

For example, you may have a critical condition in which the condition is unacceptable for any node, or you may want to consider the condition a violation only if the condition is true for 50% or more of the nodes in a tier.

Options for this evaluation scope are:

- The tier average: Evaluation is performed on the tier average instead of the individual nodes.
- Any node: If any node exceeds the thresholds, the rule is violated.
- Percentage of the nodes: If x% of the nodes exceed the thresholds, the rule is violated.
- Number of nodes: If x nodes exceed the thresholds, the rule is violated.
Health Rule Management

On this page:
- View Health Rule Status in the UI

Related pages:
- Health Rules
- Configure Health Rules
- Troubleshoot Health Rule Violations
- Policies
- Actions

To view current health rules, including the default health rules, and to access the health rule wizard, click Alert & Respond > Health Rules. Then choose the type of entity for which you want health rules from the pulldown menu at the top.

Current health rules are listed in the left panel. If you click one of these rules, a list appears in the right panel showing which entities this selected health rule affects and what the status of the latest evaluation is. You can also select the Evaluation Events tab to see a detailed list of evaluation events.

In the left panel, you can directly delete or duplicate a health rule. From here you can also access the health rule wizard to add a new rule or edit an existing one.

You can turn off evaluation of all health rules in the selected entity by clearing the Evaluate Health Rules check box. Check it when you want health rule evaluation to start again.

See Configure Health Rules for details on using the health rule wizard.

View Health Rule Status in the UI

Across the UI, health rule status is color-coded:
- Green is healthy
- Yellow/orange is a warning condition
- Red is a critical condition
- Grey indicates that the status of the health rule is unknown (for example, if the Controller cannot gather the data necessary to evaluate the rule)

If you see a health rule violation reported in the UI, you can click it to get more information about the violation.

Here are the health summary bars on the built-in dashboards:

**Business Transaction Health**

0 critical, 0 warning 46 normal

**Node Health**

7 critical, 0 warning 1 normal

**Servers**

0 critical, 7 warning 0 normal

A health column is displayed in various lists, such as the tier list below:
In the dashboards, health rule violations are displayed in the Events panel.
Configure Health Rules

On this page:
- Permissions
- Structure of the Health Rule Wizard
- Create and Configure a Health Rule

Related pages:
- Health Rules
- Configuration Import and Export API
- Metric Browser

This topic describes the detailed steps for configuring health rules using the health rule wizard. For more information on these settings, see Health Rules.

Permissions

To create, edit, or delete health rules, you need the Configure Health Rules permission. For more information, refer to Application Permissions.

Structure of the Health Rule Wizard

The health rule wizard contains four panels:

- Overview: Sets the health rule name, enabled status, the health rule schedule, evaluation period of the health rule data, and wait time post violation.
- Affected Entities: Sets the entities evaluated by the health rule. The options presented vary according to the health rule type you have defined.
- Critical Criteria: Sets the conditions, whether all or any of the conditions need to be true for a health rule violation to exist, and the evaluation scope—business transaction and node health policies defined at the tier level only—it also includes an expression builder to create complex expressions containing multiple metrics.
- Warning Criteria: Settings are identical to Critical Criteria, but configured separately.

You can navigate among these panels using the Back and Next buttons at the bottom of each panel or by clicking the panels in the wizard. You should configure the panels consecutively because the configuration of the health rule type determines the available affected entities in the Affected Entities panel as well as the available metrics in the Criteria panels.

Important

You must configure the panels consecutively because the configuration of the health rule type determines the available affected entities in the Affected Entities panel as well as the available metrics in the Criteria panels.

Create and Configure a Health Rule

The following table outlines the steps required to create and configure a health rule:

<table>
<thead>
<tr>
<th>Step</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Create a health rule.</td>
</tr>
<tr>
<td>2</td>
<td>Configure affected entities.</td>
</tr>
<tr>
<td>3</td>
<td>Configure health rule evaluation criteria.</td>
</tr>
<tr>
<td>4</td>
<td>Create and configure Conditions.</td>
</tr>
</tbody>
</table>
Create a Health Rule

On this page:
- Access the Health Rule Wizard
- Configure Health Rule Details

Related pages:
- Health Rules
- Configuration Import and Export API
- Metric Browser

You can use the default health rules provided by AppDynamics as-is or modify them to map to your requirements or define a custom health rule.

Access the Health Rule Wizard

1. Click Alert & Respond in the menu bar.
2. Click Health Rules either in the right panel or the left navigation pane.
3. Select the context for the health rule from the pulldown menu.
4. Do one of the following:
   - To create a new health rule, click the + icon.
   - To edit an existing health rule, select the health rule and click the Edit (pencil) icon.
   - To remove an existing health rule, select the health rule and click the Delete icon.

Configure Health Rule Details

You configure health rule details in the Overview panel.

1. Enter a name. If a name already exists, you can change it.
2. Check Enabled to enable the rule, clear the checkbox to disable it.
3. The Always option is pre-selected in the When is the rule enabled? drop-down list. If the health rule is enabled only at certain times, select other predefined schedules from the When is the rule enabled? drop-down list.
4. Click the drop-down list and select a number between 1 and 360 minutes. The value you specify is the latest time interval during which data is collected to determine if there is a health rule violation. This value applies to both critical and warning criteria. See Health Rule Evaluation Window.
5. In the Wait Time after Violation field, enter the number of minutes to wait before re-evaluating the rule for the same affected entity in which the violation occurred. See Health Rule Wait Time After Violation.
6. Save your configuration.

Create and Manage Health Rule Schedules

1. In the Overview panel of the Create Health Rule wizard, click Manage Health Rule Schedules. The Manage Health Rule Schedules window lists all the predefined time intervals.
2. To create a new health rule schedule:
   a. Click the + icon. The Create New Policy Schedule window is displayed.
   b. Enter a name for the schedule.
   c. Enter an optional description of the schedule.
   d. Enter the start and end times for the schedule as cron expressions. For example, the following custom schedule specifies a start time value of 0 0 13 ? * 2–6 and end time of 0 0 15 ? * 2–6, directing the health rule to be
evaluated from 1 pm to 3 pm, Monday through Friday:

For additional examples, you can select a predefined schedule in the Manage Health Rule Schedules window and click the Edit icon to see the cron expression for the predefined schedule. The Controller cron expressions are evaluated in PDT for SaaS controllers, and their format is based on Quartz Scheduler cron expressions. For on-premises controllers, cron expressions are evaluated according to controller time zone. For more information, see Quartz Scheduler documentation.

3. To edit a predefined schedule for health rule evaluation:
   a. Select the schedule and click the Edit icon.
   b. In the Edit Policy Schedule window, make necessary changes.
   c. Click OK to save your changes.

4. Save your configuration.

To delete a health rule evaluation schedule, select the schedule in the Manage Health Rule Schedules window and click Delete—the minus icon at the top. Click OK to confirm the deletion.
Configure Affected Entities

The **Affected Entities** panel lets you define what entities your health rule affects. The health rule type you select determines the metrics that are offered for configuration in subsequent panels of the health rule wizard. To define the affected entities:

1. Select a health rule type from the drop-down list. Depending on the type of the health rule, you can configure the corresponding entities that are affected. See Entities Affected by a Health Rule for information about the types of entities that can be affected by the various health rule types.
2. Use the drop-down list to select the entities affected by this health rule.
3. If you select entities based on matching criteria, specify the matching criteria. For example, if you select the Tier/Node Health - Transaction Performance as the health rule type, and if the health rule affects the nodes, you can restrict the health rule evaluation on the types of nodes or criteria such as meta-info, environment variables, and JVM system environment properties. Meta-info includes key-value pairs for:
   - key: supportsDevMode
   - key: ProcessID
   - key: appdynamics.ip.addresses
   - any key passed to the agent in the appdynamics.agent.node.metainfo system property
Configure Health Rule Evaluation Criteria

After configuring the entities affected by the health rule, you must define the evaluation criteria. The high-level process for configuring the criteria is:

1. Determine the type of evaluation criteria for the health rule:
   - Critical Criteria
   - Warning Criteria

   Though the configuration processes for critical and warning conditions are identical, critical conditions are evaluated before warning conditions. If you have defined a critical condition and a warning condition in the same health rule, the warning condition is only evaluated if the critical condition is not true.

   You can copy the settings between Critical and Warning condition panels and edit the fields, if required. For example, if you have already defined a critical condition and you want to create a warning condition that is similar, in the Warning Condition window click **Copy from Critical Condition** to populate the fields with settings from the Critical condition.

2. Determine the number and kind of metrics the health rule should evaluate. For each performance metric you want to use, create a condition using one of the following methods:
   - Use a single metric
   - Use a complex metric expression

   For more information, see **Configure a Condition**.

3. If you have defined multiple conditions, check if the health rule violates by selecting one of the following options:
   - All - all conditions evaluate to true
   - Any - one of the conditions evaluates to true
   - Custom - if a boolean expression comprising all conditions evaluate to true

4. For business transaction performance health rules and node health rule types that specify affected entities at the tier level, decide how many of the nodes must be violating the health rule to produce a violation event. See **Health Rule Evaluation Scope**.
Create and Configure Conditions

On this page:
- Create a Condition
- Configure a Condition
- Create a Custom Boolean Expression
- Delete a condition

Related pages:
- Health Rules
- Configuration Import and Export API
- Metric Browser

Define a condition or a set of conditions to evaluate the performance metrics of your application. Use the following options to evaluate the conditions:

- **expression builder** embedded in the health rules wizard to create a condition based on a complex expression comprising multiple interdependent metrics.
- **custom boolean expression** to evaluate multiple conditions within a health rule. You can use the **AND** and **OR** operators in a boolean expression.

Create a Condition

1. In the **Critical Condition** or **Warning Condition** window, click **+ Add Condition** to add a new condition component. A row defining the condition is displayed.
2. **Configure the Condition as required.**
3. **Continue to add conditions as required.** You can add a maximum of 8 conditions. Conditions are designated as A, B, C, and so on.
4. **From the drop-down list above the conditions, select:**
   - **All** - if all of the conditions must evaluate to true to constitute rule violation.
   - **Any** - if any of the conditions must evaluate to true to constitute rule violation.
   - **Custom** - if a combination of conditions defined in a boolean expression must evaluate to true to constitute rule violation. For more information on how to create a custom boolean expression, see **Create a Custom Boolean expression**.
5. **For health rules based on the following health rule types:**
   - Business Transaction
   - Node health-hardware
   - Node health-transaction performance

You must specify the evaluation scope in the **Critical Criteria** and **Warning Criteria** panels:

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

- The BT Average (average for all Nodes in the BT)
- Any Node
- % of the Nodes
- of the Nodes

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

**Evaluating Serverless Tiers**

When you monitor serverless entities comprising tiers for AWS Lambda, the health rules are evaluated as described below.
<table>
<thead>
<tr>
<th>Health Rule Type</th>
<th>Affected Entities</th>
<th>Condition Evaluation Criteria</th>
<th>Evaluation</th>
</tr>
</thead>
</table>
| • Business transactions  
• Service end points  
• Error rates | serverless tier(s) | The BT Average | Metrics are aggregated at the tier level. |
| | serverless node(s) | • Any node  
• % of the Nodes  
• Number of the Nodes | Metrics for serverless tiers are aggregated at the tier level, while the metrics for other tiers are evaluated as per the defined criteria. |
| Tier/Node Health (Transaction Performance) | serverless tier(s) | • The Tier Average  
(average for all Nodes in the Tier)  
• Any node  
• % of the Nodes  
• Number of the Nodes | Metrics for serverless tiers are aggregated at the tier level, regardless of the evaluation criteria defined. |
| | serverless node(s) | • The Tier Average  
(average for all Nodes in the Tier)  
• Any node  
• % of the Nodes  
• Number of the Nodes | The performance of serverless tiers is not evaluated for Tier/Node Health (Hardware) health rules. AWS does not offer node-level dashboards or metrics because the serverless platform runtime instances spin up and down on-demand. |
| Tier/Node Health (Hardware) | • serverless tier(s)  
• serverless node(s) | - | The performance of serverless tiers is not evaluated for Tier/Node Health (Hardware) health rules. AWS does not offer node-level dashboards or metrics because the serverless platform runtime instances spin up and down on-demand. |

**Configure a Condition**

1. In the first field of the condition row, enter a name for the condition.  
   This name is used in the generated notification text and in the AppDynamics console to identify the violation.
2. From the drop-down list below the **Add Condition** button, define metrics to evaluate the condition, select:
   - **Single metric to specify a simple metric.**
     a. From the Value drop-down list, select a qualifier for the metric from the following options:

<table>
<thead>
<tr>
<th>Qualifier Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum</td>
<td>The minimum value reported across the configured evaluation time length. Not all metrics have this type.</td>
</tr>
<tr>
<td>Maximum</td>
<td>The maximum value reported across the configured evaluation time length. Not all metrics have this type.</td>
</tr>
</tbody>
</table>
   | Value          | The arithmetic average of all metric values reported across the configured evaluation time length.  
This value is based on the type of the metric. |
   | Sum            | The sum of all the metric values reported across the configured evaluation time length. |
   | Count          | The number of times the metric value has been measured across the configured evaluation time length. |
   | Group Count    | The number of nodes contributing to a metric value, generally relevant for application or tier level metrics. |
   | Current        | The value for the current minute. |
b. To specify a simple metric, click **Select a Metric**. Metric Selection window is displayed. The metric browser in the **Metric Selection** window displays metrics appropriate to the health rule type. Alternately, you can define a relative metric path.

c. Select a metric to monitor and click **Select Metric**.

or

• Metric Expression to build a metric expression

3. From the drop-down list after the metric, select the type of comparison to evaluate the metric.

• To limit the effect of the health rule to conditions during which the metric is within a defined range—standard deviations or percentages—from the baseline, select **Within Baseline** from the menu. To limit the effect of the health rule to when the metric is not within that defined range, select **Not Within Baseline**. Then select the baseline to use, the numeric qualifier of the unit of evaluation and the unit of evaluation. For example:

```
Within Baseline of the Default Baseline by 3 Baseline Standard Deviations
```

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

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

• To compare the metric with a baseline, select `< Baseline` or `> Baseline` from the drop-down list, and then select the baseline to use, the numeric qualifier of the unit of evaluation and the unit of evaluation. You can use the Baseline Standard Deviation or Baseline Percentage as the unit of evaluation. For example:

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

See **Dynamic Baselines** for information about the baseline options.

**Baseline Percentages**

The **baseline percentage** is the percentage above or below the established baseline at which the condition will trigger. For example, if you have a baseline value of 850 and you have defined a baseline percentage of > 1%, the condition is true if the value is > \[850+\{850\times0.01\}\] or 859.

To prevent health rule violations from being triggered when the sample sets are too small, these rules are not evaluated if the load—the number of times the value has been measured—is less than 1000. For example, if a very brief time slice is specified, the rule may not violate even if the conditions are met, because the load is not large enough.

4. If you want the condition to evaluate to true whenever a configured metric does not return any data during the evaluation time frame, check the **Evaluate to true on no data** option.

This option does not affect the evaluation of unknown in the case where there is no enough data for the rule to evaluate. For example, if the health rule is configured to evaluate the last 30 minutes of data and a new node is added, the condition evaluates to unknown for the first 30 minutes even if the **Evaluate to true on no data** box is checked.

5. If you want to define a ‘Persistence Threshold’ for the condition to reduce false alerts:

a. Select ‘Trigger only when violation occurs ___ times in the last ___ min(s)’

b. Define the number of times metric performance data should exceed the defined threshold to constitute a violation
c. If required, adjust the evaluation time frame by setting an alternate evaluation time frame

You can define a persistence threshold for a condition only if you have defined the evaluation time frame of 30 minutes or less.

6. Click Save when done.

Using Health Rule Conditions to evaluate agent availability metrics can result in false positives. For example:

- Agents may not be connecting with controllers due to communication errors for a couple of minutes.
- Data may be delayed for a couple of minutes due to latency issues.

You can avoid occasional one-to-two minute metric loss due to network issues or late arrival by configuring your Health Rule as follows:

1. Select Nodes that the health rule affects. You can also set Tiers, however, it is recommended that you set Nodes.
2. Select Node Health - Hardware, JVM, CLR as the health rule Type.
3. Use the last five minutes, with a wait time of ten minutes.
4. Set your condition to be the Sum of $< \text{Specific value of three}$.

This configuration generates a violation when the agent is down for more than two minutes during the last five minutes.

**Build an expression**

To access the expression builder to create a complex expression as the basis of a condition, select the Metric Expression option from the drop-down list and click Add Expression. The Metric Expression window is displayed that allows you to construct a mathematical expression to use as a metric.

For example, the following expression is created to measure the percent of slow business transactions. See the screenshot that follows for the UI location where each step is performed.

1. In the Variable Declaration pane of the Mathematical Expression builder, click + Add variable to add a variable.
2. In the Variable Name field enter a name for the variable.
3. From the drop-down list, select the qualifier for the metric from the following options:

<table>
<thead>
<tr>
<th>Qualifier Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum</td>
<td>The minimum value reported across the configured evaluation time length. This qualifier is not available for all the metrics.</td>
</tr>
<tr>
<td>Maximum</td>
<td>The maximum value reported across the configured evaluation time length. This qualifier is not available for all the metrics.</td>
</tr>
<tr>
<td>Value</td>
<td>The arithmetic average of all metric values reported across the configured evaluation time length. This value is based on the type of the metric.</td>
</tr>
<tr>
<td>Sum</td>
<td>The sum of all the metric values reported across the configured evaluation time length.</td>
</tr>
<tr>
<td>Count</td>
<td>The number of times the metric value has been measured across the configured evaluation time length.</td>
</tr>
<tr>
<td>Group Count</td>
<td>The number of nodes contributing to a metric value, generally relevant for application or tier level metrics.</td>
</tr>
<tr>
<td>Current</td>
<td>The value for the current minute.</td>
</tr>
</tbody>
</table>

4. Click Select a metric to open an embedded metric browser.

**Health Rule Evaluation Condition**

A health rule is not evaluated if any metric in the expression has a null value. This is to avoid erroneous evaluations as shown in the following examples:

<table>
<thead>
<tr>
<th>Expression</th>
<th>Null Value</th>
<th>Evaluation</th>
</tr>
</thead>
</table>

4. Repeat steps 1 through 4 for each metric that you use in the expression. You can remove a variable by clicking the delete icon.

5. In the Expression pane, build the expression by clicking Insert Variable to insert variables created in the Variable Declaration pane along with appropriate mathematical signs.

6. When the expression is built, click Save.

Create a Custom Boolean Expression

Once you define all the conditions required for the health rule, you can create a custom boolean expression to evaluate the health rule.

1. From the drop-down list above the conditions, select Custom option.
2. Enter a combination of conditions using AND and/or OR operators. For example, (A OR B) AND C.
3. Click Save.

Modify the Custom Boolean Expression

1. Select the expression in the Condition Combination field.
2. Edit the boolean expression as required.
3. Click Save.

Delete a condition

Delete a condition component by clicking the delete (X) icon.

If you delete a condition, update the boolean expression accordingly.
Define Custom Metrics for Multiple Entities

On this page:
- To get the relative metric path for a multi-entity metric:
- To configure a health rule that evaluates the custom metric over multiple entities

Related pages:
- Health Rules
- Configuration Import and Export API
- Metric Browser

To specify Hardware Resources, JVM, and CLR metrics in multiple entities using a wildcard, you can use the procedure described in Using Wildcards in Metric Definitions.

To create a health rule on a custom metric in a single business transaction, node, or overall application performance, you specify the health rule type as custom and when you configure the condition component, in the Select Metric window choose to Specify a Metric from the Metric Tree and select the metric from the embedded metric browser.

A different use case is to create a rule that evaluates a custom metric that exists across various entities, for example across several nodes. You want to do this with one health rule; you do not want to create a separate health rule for each node. In this case, you need to specify the custom metric using the relative metric path to the metric instead of selecting the metric from the embedded metric browser.

First get the relative path to the metric and then configure the health rule using that relative path.

To get the relative metric path for a multi-entity metric:

1. Navigate to the Metric Browser by selecting Metric Browser in the left navigation pane.
2. Select the metric that you want to use for the condition.
3. Right-click and select Copy Full Path.
4. Save this value in a file from which you can copy it later.

The following example gets the metric path for the CPU %Busy metric for the Inventory Server tier. The CPU %Busy metric would be appropriate to use in a health rule that affects all the nodes in that tier.
To configure a health rule that evaluates the custom metric over multiple entities

1. In the Overview panel of health rule wizard choose the health rule type for the kind of entity that you are monitoring.
2. In the Affected Entities panel select the affected entity.
3. When you create the condition component that uses the metric, in the Select Metric window choose Specify a Relative Path Metric.
4. Crop the relative metric path that you saved from the metric browser by doing one of the following:
   - For all health rule types except Node Health-Hardware, JVM, CLR or Custom, crop the path to use the metric name alone - for example, Average Wait Time (ms)
   - For Node Heath-Hardware, JVM, CLR and Custom health rule types, crop the path to use everything after the entity, for example, after the Node name. In the example below, the cropped path would look like this.

5. Paste the cropped relative metric path in the relative metric path field of the Select Metric window.
6. Click Select Metric.
JMX Health Rules

JMX health rules establish the health status of entities in a monitored Java application based on JMX metrics.

JMX Instance Names

For health rules based on JMX metrics, you can create health rules on a node or on an entity called a JMX instance name. If you create the health rule on the JMX instance name, you have the option of restricting it to specific nodes. If you create the health rule on the node, you have the option of restricting it to specific JMX instance names.

A JMX metric is identified by its JMX metric path, its JMX instance name, and its metric name. These are determined by the Metric Path, Object Name Match Pattern, and Metric Name fields specified in the AppDynamics JMX metric MBean configuration screens. The instance name is derived from the value returned by the object match pattern. Identical JMX instance name values are considered distinct when their metric paths differ.

JMX instance names appear in the Metric Browser under the appropriate JMX metric path. Specific JMX metrics are reported under each JMX instance name.

Creating JMX Health Rules

For details about how JMX metrics are configured, see Configure JMX Metrics from MBeans.
To create a health rule on one or more JMX metrics, in the Affected Entities panel of the health rule wizard set the type of the rule to Node Health-JMX—connection pools, thread pools, and so on.

Determine what JMX objects the health rule will be evaluated on. In some IT organizations, different teams are responsible for different MBeans. In other organizations, different teams are responsible for different nodes or tiers.

The way your organization is set up determines how you think about JMX health, especially where it intersects with node health.

- As a team member responsible for a node, do you consider your node unhealthy because one or more of its MBeans is unhealthy? If so, which nodes or how many?
- As a team member responsible for your JMX infrastructure, are you primarily interested in the health of your JMX data regardless of the nodes that use it? Or are you interested in just specific JMX metrics in specific nodes? Which ones?

It is to be noted that an Instance identifier is required when configuring the JMX health rule. You can distinguish the JMX metrics based on the Instance identifier.

For the purpose of configuring JMX health rules and using them in policies, the ultimate question is: who will receive the alert when the agent reports unhealthy performance detected by JMX metrics? The flexibility of the AppDynamics health rules lets you fine-tune your JMX health rules so that the right people are alerted for the code for which they are responsible.

Select whether the affected entity should be the JMX instance name or the node. In either case, you can configure the rule to cover one of the following scopes:

- All JMX instance names in the application
- Specific JMX instance names
- All nodes in the application
- Specific nodes in the application
- Nodes within the specified tiers
- Nodes matching a given criteria

You can limit the evaluation of the health rule to either the specified JMX instance names or nodes depending on the evaluation scope.

**JMX Health Rules Affecting a Node**

In one organization, teams are responsible for nodes. Mark is responsible for WEB1_NODE, Tao for WEB2_NODE, and so on.

If an MBean in WEB1_NODE generates JMX metrics that violate a critical condition, and a health rule is configured to evaluate a JMX object in that node, Mark or someone on Mark's team will get an alert. The configuration of the health rule would be:
If different people on Mark’s team are responsible for different MBeans used in WEB1_NODE, they could refine the rule further by selecting specific JMX instance names for evaluation. For example, the last decision of this configuration can restrict the rule to evaluate only metrics in the `jdbc/ECommerceDB` JMX instance name.

Mark’s team could create a similar rule for the remaining JMX instance names to use in a policy that alerts whoever on Mark’s team is responsible for the affected JMX instance names.

Mark’s team could also create different rules that evaluate a different set of JMX metrics by choosing a different metric path in the Select JMX Objects field. For example, they could select All Web Container Runtimes.

For problems on WEB2_NODE, they would create a different health rule with WEB2_NODE as the affected node, so that the alerts for those problems go to Tao’s team.

**JMX Health Rules Affecting a JMX Instance Name**

In another organization, teams are responsible for various parts of the JMX infrastructure. Mary is responsible for `jdbc/OracleECommerceDB`, Meera for `jdbc/ECommerceDB`, and so on, regardless of which nodes use these MBeans.

So if metrics in `jdbc/OracleECommerceDB` violate a critical condition, they want Mary to get the alert because her team is responsible. The configuration of the health rule would be:
The rule could be refined to evaluate the JMX metrics in the specified JMX instance names in all nodes in the application or only in specific nodes.

This health rule affects JMX objects of one or more nodes.

Evaluate JMX metrics in this path only.

Select JMX objects with the specified name.

Limit evaluation to specified nodes only.

The rule could be refined to evaluate the JMX metrics in the specified JMX instance names in all nodes in the application or only in specific nodes.
Troubleshoot Health Rule Violations

A health rule is violated when the health rule processor detects that the health rule's critical or warning condition is true.

In this case, a health rule violation is created with a status of Open, and a Health Rule Violation Started - Critical event or a Health Rule Violation Started - Warning event is generated.

A health rule violation ends when it is either:

- Resolved: The agent reports metrics that indicate that the violated condition is no longer true. When the violation status of a health rule becomes Resolved, a Health Rule Violation Canceled - Critical event or a Health Rule Violation Ended - Warning event is generated.
  
  or

- Canceled: The health rule processor can no longer accurately assert that the health rule violation continues to violate or that it has ended. In this case, the health rule violation status is Canceled. This can occur when:
  - The health rule is edited
  - The health rule is disabled
  - Affected entities or evaluation entities on which the health rule is based have been added or removed
  - The metric values on which the health rule violation is based have become UNKNOWN

  When the violation status of a health rule becomes Canceled, a Health Rule Violation Ended - Canceled event or a Health Rule Violation Canceled - Warning event is generated.

If the same health rule is violated after a violation of it has been resolved or canceled, a new health rule violation is started.

During the life of a single health rule violation, there may be other types of health rule violation events such as Health Rule Violation Ungraded/Downgraded/Continues events.

The figure below illustrates the health rule violation life cycle.
The boxes represent the health rules violation statuses that you see in the health rule violations list—Troubleshoot > Health Rule Violations. To get more information about a particular violation select the violation in the list and click Details. You can also view the health rule violations on the Controller UI.
Health rule violation events are listed in the Events tab of various dashboards.
Because there is a set of default health rules, you may see health rule violations reported for your application even if you have not set up your own health rules. Violations reported for the APPDYNAMICS_DEFAULT_TX business transaction are for default health rule violations in the All Other Traffic business transaction.

**Find Health Rule Violations**

To find all health rule violations:

1. In the left navigation pane, click **Troubleshoot > Health Rule Violations**. The list of health rule violations displays.
   
   You can also access this screen directly from the left navigation pane of an EUM application.

2. Select **All Health Rule Violations in the Time Range** or **Only Health Rule Violations Open Now** from the drop-down list.
   
   It is possible that health rule violations that were reported are no longer open because remedial action has been taken or performance has improved on its own.

3. You can filter the list. To see the filters, click **Filters**. To hide them, click **Filters** again.
   
   With the filters showing in the left filters panel, you can select the health rule violations that you want to troubleshoot.
You can view all health rule violations or expand the nodes in the tree to select by health rule type or affected entity, such as business transaction, tier or node.

You can filter health rule violations by entering the name of the health rule in the search field on the upper right. The health rule violations are displayed in the right panel, with their affected entity, status, description, start time, end time and duration, if ended.

Examine a Health Rule Violation

To view details about a particular health rule violation:

1. Select the health rule violation row in the list.
2. Click View Health Rule Violation Details.

In the Health Rule Violation summary window, you can click the Affects link to see the dashboard of the entity affected by the health rule. Alternately, you can click the View Dashboard During Health Rule Violation button to view the dashboard at the time the violation occurred.

You can also access details of a health rule violation from the health indicators in the UI. For example, if you see the red indicator on the dashboard indicating that Business Transaction Health or Server Health is critical, click it to get the list of business transactions or tiers and then click the icon in the Health column of the list.

View Actions Triggered by a Health Rule Violation

For health rules that trigger actions configured by policies, you can get information about the action that was executed.

To view the action triggered by a health rule violation:

1. In the dashboard of the entity affected by the health rule violation, click the Events tab.
2. In the events list, locate the health rule violation that you are interested in examining. If an action was triggered by the health rule violation event, you will see an event icon in the Actions column of the list.
3. Select the row for the event.
4. Click Details.
5. In the health rule violation window, click the Actions Executed tab.
Actions

On this page:

- Types of Actions
- Actions Limits
- Actions Requiring Approval
- Create and Modify Actions
- View Actions
- Action Suppression

Related pages:

- Action Suppression
- Configuration Import and Export API
- Install the Standalone Machine Agent
- Monitor Events

An action is a predefined, reusable, automated response to an event. You can use actions to automate your runbooks.

A policy can trigger an action in response to any event. You configure which actions are triggered by which events when you configure policies.

Types of Actions

You can create the following types of actions:

- Notification
- Diagnostic
- Remediation
- HTTP Request
- Custom

Not all actions are applicable to all application environments or to all situations. Below are some general guidelines concerning different types of actions. For more details, see the pages on the specific actions before you assign an action to a policy.

- The diagnostic thread dump actions can be performed only on nodes running a Java agent.
- The diagnostic session actions can be triggered only by violations of business transaction performance health rules or by slow or stalled transaction events since these are the events that produce a view into transaction snapshots.
- Remediation actions run a local script in a node and are available on nodes running on machines that have an installed machine agent.
- Custom Actions require a dedicated Controller, deployed using either the on-premises or SaaS option. This feature is not supported for accounts on multi-tenant SaaS controllers.

Actions Limits

The Controller limits the actions invoked based on the number of triggering events per event type. There is a maximum of ten events for any single event type that can trigger actions in a given minute. If the number of triggering events per type exceeds the limit, the actions that would have been triggered by the excess events are not started. You will not see a visual indication that these actions are not being started.

For example, your application can have up to ten Health Violation Started events triggering actions and up to ten Resource Pool Limit Reached events triggering actions within the same minute. But if you have eleven Health Violation Started events firing, the action that would be triggered by the eleventh event is not started.

To reduce unnecessary actions, there is a limit on the number of diagnostic and remediation actions that AppDynamics will invoke. The default limit is five actions per minute per machine for each type of action.

If, for example, a policy is configured on all the nodes where there are 100 nodes triggering actions, AppDynamics randomly selects five of the actions to execute.

To avoid exceeding the limits, design your policies so that they do not trigger an excessive number of actions for any particular event. You can generate fewer events by configuring the affected entities of your health rules at the tier level. See entities affected by a health
rule and by health rule types in Health Rules.

**Actions Requiring Approval**

For actions that take thread dumps or run a local script, you can optionally require email approval to run the action whenever it is triggered. If you check this option, human intervention is required before the automated action actually starts.

If you specify the approval required option when you configure the action, when the action is triggered an email containing a link is sent to the configured email address. The link presents a login screen (if the user is not already logged in to AppDynamics) and after the user logs in, a dialog requesting approval to take the thread dump or run the script. The user can click in this dialog to approve and start the action or cancel the action.

If you do not check the Require approval option before executing the Action check box, the action will start automatically with no human intervention.

**Create and Modify Actions**

To configure actions, you need the **Configure Actions** permission.

To access the actions configuration screens

1. Click Alert & Respond in the menu bar.
2. Click Actions either in the right panel or the left navigation pane.
3. Select the context for the action (Application, User Experience Browser App, Databases, Servers, Analytics) from the pulldown menu.

The list of configured actions is displayed.

You can filter the types of actions displayed in the actions list by checking the type in the pulldown filter. If no action types are checked there is no filtering, but if at least one type is checked, then only actions of that type are displayed.

4. Do one of the following:
   - To create a new action, click the Create Action (+) icon.
     After you have clicked Create Action, the instructions depend on the type of action you are creating. See the topics for the action type you have selected.
   - To edit an existing action, select the action and click the Edit (pencil) icon.
   - To remove an existing action, select the action and click the Delete (-) icon.

**View Actions**

While actions are being triggered by events, you can get a summary of their execution in the Actions column in the Events list. You can access the events list from the Events tab of the application, tier or node dashboard.

An icon in the Actions column indicates that an action was triggered for this event, the type of the icon indicating the type of action. If the action icon is grayed out, the action is either still executing or failed to complete, the tooltip indicates which. If the icon is fully displayed, the action executed successfully.

**Action Suppression**

You can prevent policies from firing actions for a configurable time period.
Notification Actions

On this page:

- Email Notifications
- SMS Notifications

Related pages:

- Alert and Respond
- Policies
- Actions
- Email Templates
- Enable an Email Server

A notification action sends an email or SMS to a recipient or recipient list.

If you are using a SaaS Controller, all notification timestamps are in Pacific time.

Email Notifications

You can create an email notification with or without using an email template.

Without a Template

If you create an email notification without a template, you provide only the email address of the notification recipient. The name of the action is the name of the recipient. The contents of the email are automatically generated by the policy that triggers the action. The generated message contains a link directly to the Controller screen that details the triggering event, which is the place for the recipient to start troubleshooting the problem.

This is what the message with the deep link looks like if you create an email notification without using a template:

![Email Notification Example](image)

When you configure a policy to fire an email notification action that has not been created with a template, you have an opportunity to add a note to the email at that time. This note is applied only when the action is invoked by the particular policy. By adding an optional note, you can customize email notifications for the policies that invoke them.
With a Template

If you create an email notification with a template, you provide the name of the action and the name of the email template to use. The template provides for addressing multiple recipients and you can add and delete recipients when you create an action using the template.

The template must already exist before you can use it in an action. Email templates are created by users who have account-level permissions to create templates. See Email Templates for information about creating templates.

The body of the message is specified in the email template. It is not generated automatically.

All the text in the message must be created in the template. Customization is accomplished through the use of variables in the message body, which are replaced by actual values when the message is sent.

You might want to use a template:

- to reuse the notification in other policies
- to customize the body of the message
- to decorate the notification with your own branding
- to add custom SMTP headers to the email
- to control whether the email is sent once or several times for each event
- to control the maximum number of triggering events listed in the email using a clamp limit
- to integrate with third-party email APIs, such as Remedy

To create an email notification:

1. Access the Create Action window. See Create and Modify Actions in Actions.
2. Under Notifications, select Send an email in the Create Action window.
   If you want to use a template to create the email notification, check Use template?
3. Click OK.
4. Do one of the following:
   - If you are not using a template, enter the email address to which to send the notification.
   - If you are using a template, name the action and select the appropriate template from the dropdown list.
     Unless disallowed by the template, you can add recipients at this time but you cannot remove any of the required recipients that were configured in the email template.
5. Click Save.

SMTP host configuration for email template

If email and SMS settings have not been configured for AppDynamics, configure them now. See Configure the SMTP Server.
SMS Notifications

The content of the SMS is automatically generated. It contains:

- the notification header
- the application name
- the triggered time

Notifications of health rule violations also include:

- name of health rule violated

Event notifications also include:

- event notification configuration name
- map of event types to the number of these events

An SMS notification configuration specifies the phone number of the recipient.

To create an SMS notification:

1. Access the actions configuration window. See Create and Modify Actions in Actions.
2. Under Notifications, select Send an SMS message in the Create Action window.
3. Enter the phone number to send the notification.
4. Click OK.

If email and SMS settings have not been configured for AppDynamics, configure them now. See Enable an Email Server.
Email Templates

On this page:
- Permissions
- Access Email Templates
- Custom Templating Variables
- Admin Email Recipients
- Email Template
- Custom Email Headers
- One Email Per Event
- Event Clamp Limit
- Test Email Template

Related pages:
- Notification Actions
- Roles and Permissions
- Predefined Templating Variables

Email templates can be optionally used by Email notification actions.

Permissions
The account-level Configure Email Templates permission is required to create an email template. See the Alert and Respond topic for information about permissions related to Alert and Respond features.

Access Email Templates
To access existing templates or to create new ones, in the application click Alert & Respond in the menu bar, then Email Templates either in the right panel or the left navigation pane.

The Email Templates list appears. Click an existing template in the list to view, edit or delete it. Click the New + icon to create a template.

Use the new template dialog to define the template. While the configuration settings are largely self-explanatory, some template configuration features and options may require additional information. The following sections describe those features.

After you have created the template, click the Save button at the bottom.

Custom Templating Variables
You can use variables that replace values in the message body when the email is sent.

The template already knows a set of predefined variables, which are described in Predefined Templating Variables. Check this list before you create any custom templating variables. Chances are the variable you want to use has already been defined.

You can optionally configure custom variables if the predefined variables do not meet all your needs. When a predefined variable and a custom variable are both configured with the same variable name, the template uses the predefined variable.

The template uses Apache Velocity version 1.7 to process the variables. See the Velocity User Guide for details about usage.

Admin Email Recipients
For each recipient enter an email address and click + Add. To remove a recipient click -.

If the Allow custom email recipients check box is checked in the template, users of the email template can add additional email recipients when they create the email notification, but they cannot delete any of the recipients configured by the template. Clear this check box if you do not want to allow users of the template to be able to modify the recipient list.

The only way to remove a required recipient is by modifying the template.

Email Template
Enter the subject line and the body of the message. AppDynamics recommends including both an HTML version and a plain text version. You can use any predefined or custom templating variables in both the HTML and plain text message bodies.
**d-color** instead of **background** for HTML email templates.

For-each loops are supported.

See **Predefined Templating Variables** for an example.

**Custom Email Headers**

Use this section to add custom SMTP headers to the message.

**One Email Per Event**

Several separate events, or separate occurrences of the same event, could potentially invoke the same notification action.

The One Email Per Event setting controls whether the action bundles the emails triggered by those events. The effect is:

- If this check box is clear, the email is sent once every minute, no matter how many events triggered the notification within that minute. This is the default behavior.
- If this check box is checked, email is sent every time an event triggers it. In this case, if ten events trigger the action, the email is sent ten times, even if all those events occurred within a single minute.

**Event Clamp Limit**

If you have not checked the One Email Per Event check box, you might want to limit the display of the events that triggered the action in the email, especially if the potential list of events could be long. The clamp limit is the number of most recent triggering events to be shown in the email. This setting is disabled if One Email Per Event is checked.

**Test Email Template**

After you save the email template, you can test the template by sending an email.
Test Email Templates

After you save the email template, you can test the template by sending an email.

1. In the Create Email Template dialog box, click **Test**. The Email Template Test dialog box is displayed.
2. In the Email Template Test dialog box, specify the test variables to use for the test. These variables may be different from those that will be used in the real emails that are sent automatically.
3. Set the log level or the amount of detail that you want to see: INFO, WARN, ERROR, TRACE, and DEBUG. The log level defaults to DEBUG, but you can change this using the dropdown Log Level list.
4. Specify the type of events that trigger the email action along with the count. You can add more triggers if required.
5. Click **Save**.

**Run Test**

When you run the test, an email is sent when the action is triggered. You get a transcript of the test.
If the results are not what you expect, you may need to modify the email template, the event triggers, or both before you use the template in an action.
Diagnostic Actions

A diagnostic action can:
- Start a diagnostic session to collect snapshots
- Take a thread dump, Java only

When performance is slow or your application is experiencing a lot of errors you can start a diagnostic action to get to the root cause.

A diagnostic session gives you a view into captured transaction snapshots with full call graphs. These snapshots help you diagnose violations of business transaction performance health rules or slow or stalled transaction events. The affected entity of the event triggering a diagnostic session must be a business transaction.

A thread dump is a general-purpose snapshot of the state of all threads that are part of a JVM process. The state of each thread is presented with a stack trace that shows the contents of each thread's stack. Thread dumps are used for diagnosing JVM performance problems, such as code deadlocks.

Diagnostic Session Actions

A diagnostic session is always associated with a business transaction. It shows transaction snapshots with full call graphs to help you drill down to the root cause of a problem.

To create a diagnostic session action:

1. Access the actions configuration window. See Create and Modify Actions in Actions.
2. Under Diagnostics, select Start a Diagnostic Session on the selected Business Transactions in the Create Action window and click OK.
3. After entering a name for the action, the duration of the diagnostic session in minutes and the number of snapshots to take per minute, select whether a diagnostic session will be started for any business transaction affected by the event or for specific business transactions.
   If you choose specific business transactions, specify the business transactions that will trigger the diagnostic session by moving them from the available list to the selected list. The business transactions that you can specify are not limited to those that triggered the action.
4. Click OK.

Diagnostic Action Results

The results of a diagnostic action that has executed are available in the events list for the event that triggered the action.

To get the details of a diagnostic session or a thread dump that has been initiated by an action:

In the Events list, locate the row for the event that triggered the action for which you want to see the results. In the Actions column:

- Click this icon to see the details of a thread dump action:
Click this icon to see the details of a diagnostic session action:

On disk, the thread dumps are stored in the `app_agent_operation_logs` directory in the Controller home. The files are named based on the ID in the `app_agent_operation` table.

Threads identified by AD Thread are threads initiated by AppDynamics app agent code.

**Thread Dump Actions**

You can direct the Java Agent to take a thread dump for a specified number of samples (maximum of 50) with each sample lasting for a specified number of milliseconds—maximum of 500 ms. The thread dump is executed on the node monitored by the agent.

Users need the *Agent Advanced Operation* permission to request a thread dump.

**Agent Limit on Thread Dumps**

One thread dump operation is executed at a time. They are not executed in parallel. If additional thread dump requests are received while one is being executed, they are queued with a limit of five per agent.

If the five thread dumps per agent limit is exceeded, the console shows an event with a thread dump operation that was skipped because of the limit and the associated action dialog for the executed policy links to this event.

**To create a thread dump action:**

1. Access the Create Action window.
2. Under Diagnostics, select **Take a thread dump** in the Create Action window and click **OK**.
3. Enter a name for the action, the number of samples to take and the interval between the thread dumps in milliseconds. If you want to mandate an approval before the thread dump action can be started, check the **Require approval before executing this Action** check box and enter the email address of the individual or group that is authorized to approve the action. See **Actions Requiring Approval** for more information.
4. Click **OK**.

When a thread dump action is triggered by a backend discovery event, if the backend is not resolved quickly the policy will not start the thread dump.
Remediation Actions

On this page:
- Prerequisites for Local Script Actions
- Remediation Scripts
- Remediation Example
- Create a Local Script (Remediation) Action

Related pages:
- Actions
- Policies
- Remediation Scripts
- Install the Standalone Machine Agent

A remediation action runs a local script in a node. The script executes on the machine from which it was invoked or on the node specified by the remediation action configuration. You can use this type of action to automate your runbook procedures. You can optionally configure the remediation action to require human approval before the script is started. See Actions Requiring Approval.

Prerequisites for Local Script Actions

- The Standalone Machine Agent must be installed running on the host on which the script executes. To see a list of installed machine agents for your application, click View machines with machine-agent installed in the bottom left corner of the remediation script configuration window.
- To be able to run remediation scripts, the machine agent must be connected to an on-premises Controller or to a SaaS Controller via SSL. Remediation script execution is disabled if the machine agent connects to a SaaS Controller on an unsecured (i.e., non-SSL) HTTP connection.
- The Standalone Machine Agent OS user must have full permissions to the script file and the log files generated by the script and/or its associated child processes.
- The script must be placed in `<agent install directory>\local-scripts`.
- The script must be available on the host on which it executes.
- Processes spawned from the scripts must be daemon processes.

Remediation Scripts

A remediation script is run on the machines that you specify in the remediation script configuration. You can run the script from the machine affected by the violation that triggered the action or from a central management server. It is not necessary for an app agent to be running on the machine on which the script executes, just a machine agent.

Remediation Example

The following remediation action, named increasePool, executes a local script named `runbook.sh`, which increases the size of the connection pool on the JVM.
A policy named `ConnectionPoolPolicy` triggers this action when the Resource Pool Limit Event fires:

**Edit Policy - ConnectionPoolPolicy**

<table>
<thead>
<tr>
<th>Trigger</th>
<th>Health Rule Scope</th>
<th>Object Scope</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name: ConnectionPoolPolicy</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Enabled</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Execute actions in batch</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This Policy will fire when any of these Events occur

**Health Rule Violation Events**
- Health Rule Violation Started - Warning
- Health Rule Violation Started - Critical
- Health Rule Violation Continues - Warning
- Health Rule Violation Continues - Critical

**Other Events**
- Slow Transactions
- Code Problems
- Code Deadlock
- Resource Pool Limit Reached

Create a Local Script (Remediation) Action

*To create a remediation action:*

1. Access the Create Action window. See Create and Modify Actions in Actions.
2. Select Run a script or executable on problematic Nodes in the Create Action window and click OK.
3. After entering a name for the action, in the field that terminates the Relative path to script entry, enter the rest of the path to the executable script. Remediation scripts must be stored in a sub-directory of the machine agent installation. The sub-directory must be named `local-scripts`. The following paths are all valid:

```plaintext
${machine.agent.directory}/local-scripts/runMe.sh
${machine.agent.directory}/local-scripts/johns_scripts/runMe.sh
${machine.agent.directory}/local-scripts/ops/johns_scripts/runMe.sh
```

4. Click the `+` to enter the absolute paths of any log files that the script writes to that you want to be included in the script output.
5. Enter the timeout period for the script process in minutes.
6. If you want to mandate an approval before the script action can be started, check the `Require approval before executing this Action` check box and enter the email address of the individual or group that is authorized to approve the action. See Actions Requiring Approval for more information.

**Specify the nodes on which the action will run**

When you bind the action to a policy, you specify the nodes on which the script should execute. You can configure the number of nodes or the percentage of nodes or you can configure a specific node. This flexibility allows you to configure scripts to run from a central management server, not just the node on which the violation occurred.

In the Configure Action window of the policy actions to execute, either:

- Select **Execute Action on Affected Nodes** and the percentage of the nodes or the number of nodes on which to run the script.
  
  or

- To designate the specific node on which to run the script, select **Execute Action on Specified Node**, and select the node on which the script should run from the popup node browser. You can either save the configuration or change it to designate a different node.

**See the output of the local script**

1. In the Events list, locate the row for the event that triggered the action for which you want to see the results.
2. In the Actions column of the selected row, click the remediation script icon.
3. In the script results list, select the script output that you want and click **Download Local Script Result**.

When a remediation action is triggered by a backend discovery event, if the backend is not resolved quickly the policy will not start the local script.
Remediation Scripts

A remediation script is run on the machine that you specify in the remediation script configuration. You can run the script from the machine affected by the violation that triggered the action or from a central management server. It is not necessary for an app agent to be running on the machine on which the script executes, just a Machine Agent.

Guidelines for Remediation Scripts

By default, the script is a shell script in /bin/sh invoked with the -ex option, unless the script has a header, in which case the interpreter in the header is used. For example, if the script header is #!/bin/perl, the PERL interpreter is invoked.

A process exit code of zero indicates that the script execution succeeded. A non-zero exit code indicates that it failed.

The script should be written as generically as possible to allow it to run on any of the nodes for which it is invoked. AppDynamics exports the following environment variables to the script runtime to provide context regarding the environment and the event that triggered the action.

<table>
<thead>
<tr>
<th>Environment Variable</th>
<th>Cardinality</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>APP_ID</td>
<td>1 (or N)</td>
<td>Name of the Application</td>
</tr>
<tr>
<td>EVENT_TIME</td>
<td>1</td>
<td>Timestamp of the event</td>
</tr>
<tr>
<td>EVENT_ID</td>
<td>1</td>
<td>Event Id</td>
</tr>
<tr>
<td>EVENT_TYPE</td>
<td>1</td>
<td>type of event, such as: ERROR, APPLICATION_ERROR, APPLICATION_INFO,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>STALL, BT_SLA_VIOLATION, DEADLOCK, MEMORY_LEAK, MEMORY_LEAK_DIAGNOSTICS,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LOW_HEAP_MEMORY, ALERT, CUSTOM, APP_SERVER_RESTART, BT_SLOW, SYSTEM_LOG,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INFO_INSTRUMENTATION_VISIBILITY, AGENT_EVENT, INFO_BT_SNAPSHOT,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AGENT_STATUS, SERIES_SLOW, SERIES_ERROR, ACTIVITY_TRACE,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OBJECT_CONTENT_SUMMARY, DIAGNOSTIC_SESSION,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HIGH_END_TO_END_LATENCY, APPLICATION_CONFIG_CHANGE,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>APPLICATION_DEPLOYMENT, AGENT_DIAGNOSTICS, MEMORY, LICENSE</td>
</tr>
<tr>
<td>ENV_STARTUP_ARGS</td>
<td>1</td>
<td>Process args</td>
</tr>
<tr>
<td>ENV_SYSTEM_PROPERTIES</td>
<td>1</td>
<td>JVM System Props. when Java</td>
</tr>
<tr>
<td>AFFECTED_ENTITY</td>
<td>1</td>
<td>Affected Entity that triggered the event</td>
</tr>
</tbody>
</table>

Remediation scripts must be stored in a sub-directory of the machine agent installation. The sub-directory must be named local-scripts. The following paths are all valid:
Troubleshooting Remediation Scripts

To troubleshoot your remediation script, look for the process in the machine agent log. The log is located at

<machine_agent_home>/logs/machine-agent.log

The snippet below from the machine agent log shows both error and success messages from running a local script named `script.sh`.

```
{agent-scheduler-1} 07 May 2013 18:20:24.580 ERROR RunLocalScriptEventListener: Error occurred while executing run local script operation
{agent-scheduler-1} 07 May 2013 18:20:24.580 INFO RunLocalScriptRequestHandler - Received run local script request: opid=27,
actionId=61f811f8-3a38-477e-9abc-0338b547722
{agent-scheduler-1} 07 May 2013 18:20:24.991 ERROR ScriptExecutor - Script '/Users/akilman/script.sh' must reside in
'/Users/akilman/Work/cart-tmp/machineagent/local-scripts'
{agent-scheduler-1} 07 May 2013 18:20:24.991 ERROR RunLocalScriptEventListener: Error occurred while executing run local script operation
{agent-scheduler-1} 07 May 2013 18:20:24.994 INFO RunLocalScriptRequestHandler - Received run local script request: opid=28,
actionId=45202ab0-6f43-4f7a-9308-12ce722d88a2
{agent-scheduler-1} 07 May 2013 18:20:24.994 ERROR ScriptExecutor - Script '/Users/akilman/script.sh' must reside in
'/Users/akilman/Work/cart-tmp/machineagent/local-scripts'
{agent-scheduler-1} 07 May 2013 18:20:24.994 ERROR RunLocalScriptEventListener: Error occurred while executing run local script operation
{agent-scheduler-1} 07 May 2013 18:20:24.997 INFO RunLocalScriptRequestHandler - Received run local script request: opid=29,
actionId=5b8b8d8b-7859-402f-927e-643fe1b456d0
{agent-scheduler-1} 07 May 2013 18:20:24.999 INFO RunLocalScriptEventListener: Run local script request completed successfully
{agent-scheduler-1} 07 May 2013 18:20:25.582 INFO RunLocalScriptEventListener: Run local script request completed successfully
{agent-scheduler-1} 07 May 2013 18:20:25.884 INFO RunLocalScriptEventListener: Run local script request completed successfully
{agent-scheduler-1} 07 May 2013 18:26:28.249 INFO RunLocalScriptRequestHandler - Received run local script request: opid=90,
actionId=8cde8b30-184a-40da-b679-1cf28a915104
{agent-scheduler-1} 07 May 2013 18:26:28.913 INFO RunLocalScriptEventListener: Run local script request completed successfully
```
HTTP Request Actions and Templates

On this page:
- Required Permissions
- Access HTTP Action Templates
- Create or Modify an HTTP Request Template
- Test the Template

Related pages:
- Policies
- Actions
- Configuration Import and Export API
- Alert and Respond
- Predefined Templating Variables

An HTTP request action sends an HTTP request in response to an event. These types of actions allow you to integrate AppDynamics' policies with third-party HTTP APIs.

You create an HTTP request action using an HTTP request template. The template describes the HTTP request and is reusable by different HTTP request actions within an AppDynamics account.

After you create a template, users in the Controller UI can create actions that use it. The template appears as an option after the user chooses the Make an HTTP Request option in the Create Action window and access the configure action screen.

Required Permissions

To create or modify HTTP Request Templates, users need the account-level Configure HTTP Request Templates permission.

Access HTTP Action Templates

To access existing templates or to create new ones. Click Alert & Respond in the menu bar, and then HTTP Request Templates. From there, you can:
- Click an existing template from the HTTP Request Templates list to view, edit or delete it, or
- Click the New + icon to create a template.

Create or Modify an HTTP Request Template

When creating or modifying templates, click the info icons in the template to get information about how to complete the configuration. The following subsections clarify some details in the template configuration that may require additional information.

After you have created a template, click the Save button at the bottom to save it.

Custom Templating Variables

You can use variables that replace values in the URL path and payload when the HTTP request is sent. Foreach loops are supported.

The template already knows a set of predefined variables, which are described in Predefined Templating Variables. Check this list before you create any custom templating variables. Chances are the variable you want to use has already been defined.

You can optionally configure custom variables if the predefined variables do not meet all your needs. When a predefined variable and a custom variable are both configured with the same variable name, the template uses the predefined variable.

The template uses Apache Velocity version 1.7 to process the variables. See the Velocity User Guide for details about usage.

Request URL

Enter the URL for the request in the Raw URL field and select the URL encoding scheme from the pulldown menu. Only UTF-8 and ISO_8858-1 are supported. Verify that the Encoded URL is the exact request to send.
Authentication

You can define basic authentication for your HTTP request. Select BASIC authentication type from the Type drop-down list only if the communication is encrypted. If the communication is not encrypted, we recommend not to use any authentication. After selecting the authentication type, specify the authentication username and password.

Custom Request Headers

You can define custom headers for the HTTP requests. Custom Request Headers can contain custom templating variables or predefined templating variables encoded as $(VARIABLE_NAME).

For a complete list of predefined variables, see Predefined Templating Variables. Check this list before you create any custom templating variable. See the Velocity User Guide for details about usage.

Payload

To include a payload in your HTTP request, define the MIME Type from the drop-down list. You can encode the payload using UTF-8 or ISO-8859-1 encoding.

Response Handling Criteria

Success or Failure

If you do not specify any response-handling criteria for either success or failure, the HTTP response is always success.

Otherwise:

- Failure criteria are evaluated before success criteria. As soon as a match is found on a fail criterion, the response is set to failure and the remaining fail match conditions, if there are any, are not evaluated.
- If no failure criteria are matched, the success criteria are then evaluated.

If no success criteria are configured, the response will always be success as long as no fail criteria were previously matched.

Otherwise:

- If you specify at least one success criterion, there must be a match on a success criterion for the response to be success. In other words, if any success criteria are configured but not matched, the response will be failure.

Both failure and success criteria are considered in the order in which they appear in the template.

Criteria With and Without Payload

You can set separate criteria for the same status code. For example, you can set one criterion on a status with payload as success and another criterion on the same code without payload as failure.

Or, you can set separate criteria for the same status code with payload of different content types. For example, payload of type application/xml and application/xhtml+xml might be success and payload of other types could be failure.

Content Types for Payload

You can specify the content type for the payload using the dropdown list that appears for requests for which the Expect Payload checkbox is checked.

If you do not know the content type but the request expects payload, use */* to specify all content types.

The following configuration returns a success response for a 200 status code with an XML payload. Any other code will cause the
request to return a failure response.

Response Handling Criteria

Failure Criteria
Evaluate the following criteria first. Set action result to "failure" when the HTTP response matches any of the following:

<table>
<thead>
<tr>
<th>Status Code</th>
<th>Expect Payload</th>
<th>Content Type</th>
</tr>
</thead>
</table>

+ Add Failure Criteria

Success Criteria
Evaluate the following only if there is no match in the above failure criteria. Set the action result to "success" if there is a match in the following, else set it to "failure".

<table>
<thead>
<tr>
<th>Status Code</th>
<th>Expect Payload</th>
<th>Content Type</th>
</tr>
</thead>
</table>

+ Add Success Criteria

One Request Per Event

Several separate events, or separate occurrences of the same event, could potentially invoke the same HTTP action.

The One Request Per Event setting controls whether the action bundles the HTTP requests.

If this check box is clear, the HTTP request is sent once every minute, no matter how many events triggered the action within that time frame. This is the default behavior.

If this check box is checked, the request is sent every time an event triggers it. In this case, if ten events trigger the action, the HTTP request is sent ten times, even if all those events occurred within a single minute.

You might want to send one request per event, checkbox checked, if you are trying to log tickets and your ticketing system API does not support bulk create. However, if it does, you probably want a single bulk request every minute, checkbox clear.

Event Clamp Limit

If you have not checked the One Request Per Event checkbox and the list of triggering events could be large, you might want to limit the display of the events that triggered the action in the response. The clamp limit is the n most recent triggering events. The default -1 means no limit. This setting is ignored if One Request Per Event is checked.

Timeouts and Redirects

- Connect timeout: The maximum number of milliseconds to wait for the request to reach the server.
- Socket timeout: The maximum number of milliseconds to wait to receive the response.
- Max Redirects: The maximum number of times that a single request can redirect.

Test the Template

You can test the template, by clicking Test at the bottom of the template configuration.

Configure Test

In the Template Test configuration window, specify the test variables to use for the test. These variables may be different from those that will be used in the real requests that will be automatically sent when the HTTP action is live.
Also, specify the type of events that trigger for the test.

The log level defaults to INFO, but you can change this using the dropdown Log Level list.

The following sample test template configuration simulates three POLICY OPEN CRITICAL events triggering this action. More triggers could be added.

HTTP Action Template Test

HTTP Request Template http

Log Level INFO

Test Variables

var1 applications

Event Type Trigger

<table>
<thead>
<tr>
<th>Event</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slow requests - Slow</td>
<td>3</td>
</tr>
</tbody>
</table>

+ Add Event Type

Run Test

When you run the test, the request is sent when it is triggered. You get a transcript of the run.

If the results are not what you expect, you may need to modify the template test configuration, the HTTP action template, or both before you use the template in an action.
Custom Actions

On this page:
- Create a Custom Action
- Disabling Custom Actions on a Multi-Tenant Controller

Related pages:
- Build a Custom Action
- Actions

A custom action is typically used to integrate third-party alerting and ticketing systems. A custom action is different from other actions in that it executes just once on the controller instance.

The custom action is made up of a custom action script and a custom.xml file, which you must create before you can create an action that uses them. The custom action scripts include parameters for specifying the affected entity, for example, the tier, node, business transaction, and so on. See Build a Custom Action for details on how to create the custom action script and XML file.

Custom actions are commonly used when you want to trigger a human workflow or leverage an existing alerting system that is external to AppDynamics. For example, you could use a custom action to file a JIRA ticket when AppDynamics reports that a connection pool is near saturation.

Create a Custom Action

After the custom action script and custom.xml files have been tested manually and installed on the Controller and you have restarted the Controller, you can create the custom action.

To create a custom action:

1. Access the Create Action window. See Create and Modify Actions in Actions.
2. Select Run a Custom Action that has been uploaded to the Controller.
3. Click OK.
4. Enter a name for the action.
5. Select the custom action from the dropdown list. All the custom actions you have created are listed here.
6. Click Save.

The custom action is now available for assignment to a policy.

Disabling Custom Actions on a Multi-Tenant Controller

Because a custom action script has access to the Controller file system, for security reasons you may decide to disable this functionality in a multi-tenant environment.

To disable custom actions:

1. Open the Controller admin console.
2. Go to Controller Settings.
3. Search for the aas.multitenant.custom.action.local.execution.enabled property.
4. Set to false.
5. Click Save.
You can set up a custom action on the controller instance to integrate notification of AppDynamics health rule violations and events with an alerting or ticketing system. Use a push approach by creating custom notifications that pass the information to your alerting system.

**Custom Notifications and Custom Actions**

A custom notification lets you integrate alerts about AppDynamics health rule violations and events into your own alerting system. This integration extension requires:

- A `custom.xml` file that provides information about the custom notification
- An executable script that accepts parameters from AppDynamics about the events and health rule violations that trigger an alert
- Configuring AppDynamics events or policies to trigger the custom notification via a custom action

This topic describes how to create the script and the XML file. See the documentation on the Alert & Response features in the Related pages above for information on how to trigger the action.

**Creating a Custom Action**

**Create the script**

For each custom action that you want to implement, create an executable script (.bat extension for Windows, .sh extension for Linux) that can accept and process the parameters passed to it by AppDynamics. See Information Passed to the Custom Action Script from AppDynamics for details on the parameters. For each script:

- Set correct executable permissions for the shell scripts in a Linux environment. For example, `chmod 770 script1.sh`.
- Ensure that the script file has the correct character encoding. This is especially important when creating a Unix shell script on a Windows machine.

**Install the script on an On-Premises Controller**

To install the script on an on-premises controller:

1. At the top level of the Controller installation directory, create a directory named "custom” with a sub-directory named "actions".
   
   `<controller_home>/custom/actions`

2. In the `<controller_home>/custom/actions` directory, create a subdirectory for each custom action script that you will install. For example, for an action that interfaces with a JIRA system.
3. Move the script to the appropriate subdirectory that you created in step 2.

Create the XML File

1. Create a custom.xml file that describes the location and name of your custom action script. See Contents of the custom.xml File.
2. For an on-premises Controller, move the file to the `<controller_home>/custom/actions` directory. For a SaaS Controller, contact your AppDynamics sales representative for instructions.

Verify on the Script on an on-premises Controller

1. After you have installed the script and the custom.xml file, restart the Controller.
2. Verify the script manually. To verify the script:
   a. Open a command-line console on the Controller host machine.
   b. Execute the script file from the command line console.

Create the Custom Action

Create the custom action in the AppDynamics UI to arrange how the custom action will be triggered. See Custom Actions.

Contents of the custom.xml File

The custom.xml file has an `<actions>` element for every custom action on the controller.
The `<type>` element contains the subdirectory that contains the script file.
The `<executable>` element contains the name of the script.

Sample custom.xml file

```
<custom-actions>
   <action>
      <type>jira</type>
      <executable>script1.bat</executable>
   </action>
   <action>
      <type>bugzilla</type>
      <executable>script2.sh</executable>
   </action>
</custom-actions>
```

Information Passed to the Custom Action Script from AppDynamics

The custom action script must handle the parameters that the Controller passes from the health rule violation or other event. The parameter values are passed as an array of strings.

The parameters are passed as $0 for the script name, then $1, $2, ... $n. $1 is the first parameter (application name), $2 is the application id, and so on in the order in which they are documented in the sections below.
Health rule violations have a different set of parameters from events.

Parameters passed by a health rule violation

The parameters describe the violated health rule violation that triggered the action.

The total number of elements in the array depends on the number of entities evaluated by the health rule and the number of triggered conditions per evaluation entity. Examples of evaluation entities are application, tier, node, business transaction, JMX. For each evaluation entity, the script expects the entity type, entity name, entity id, number of triggered conditions, and for each triggered condition, the set of condition parameters.

The parameter values are passed in the order in which they are described below.

Structure of Parameters Sent by a Health Rule Violation

- APP_NAME
- APP_ID
- PVN_ALERT_TIME
- PRIORITY
- SEVERITY // INFO, WARN, ERROR
- HEALTH_RULE_NAME
- HEALTH_RULE_ID
- PVN_TIME_PERIOD_IN_MINUTES
- AFFECTED_ENTITY_TYPE
- AFFECTED_ENTITY_NAME
- AFFECTED_ENTITY_ID

The following parameters are passed for each evaluation entity:

- EVALUATION_ENTITY_TYPE
- EVALUATION_ENTITY_NAME
- EVALUATION_ENTITY_ID
- NUMBER_OF_TRIGGERED_CONDITIONS_PER_EVALUATION_ENTITY

The following parameters are passed for each triggered condition for this evaluation entity:

- SCOPE_TYPE_x
- SCOPE_NAME_x
- SCOPE_ID_x
- CONDITION_NAME_x
- CONDITION_ID_x
- OPERATOR_x
- CONDITION_UNIT_TYPE_x
- USE_DEFAULT_BASELINE_x
- BASELINE_NAME_x
- BASELINE_ID_x
- THRESHOLD_VALUE_x
- OBSERVED_VALUE_x
- SUMMARY_MESSAGE
- INCIDENT_ID
- DEEP_LINK_URL
- EVENT_TYPE
- ACCOUNT_NAME
- ACCOUNT_ID

Definitions of Parameters Sent by a Health Rule Violation

<table>
<thead>
<tr>
<th>Health Rule Violation Parameter</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>APP_NAME</td>
<td>Name of the business application</td>
</tr>
<tr>
<td>APP_ID</td>
<td>Application ID number</td>
</tr>
<tr>
<td>PVN_ALERT_TIME</td>
<td>Alert time, such as Thu Dec 22 15:03:56 PST 2011</td>
</tr>
<tr>
<td>PRIORITY</td>
<td>Integer designating how urgently a health rule violation should be fixed, with the lowest number (0) the most urgent</td>
</tr>
<tr>
<td>SEVERITY</td>
<td>INFO, WARN, or ERROR—in the AppDynamics UI they are called, or WARNING, and CRITICAL.</td>
</tr>
<tr>
<td>------------------------------</td>
<td>--------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>HEALTH_RULE_NAME</td>
<td>Name of the health rule that was violated</td>
</tr>
<tr>
<td>HEALTH_RULE_ID</td>
<td>Health rule ID</td>
</tr>
<tr>
<td>PVN_TIME_PERIOD_IN_MINUTES</td>
<td>Health rule violation time period in minutes</td>
</tr>
<tr>
<td>AFFECTED_ENTITY_TYPE</td>
<td>APPLICATION, APPLICATION_COMPONENT (aka Tier), BUSINESS_TRANSACTION, APPLICATION_DIAGNOSTIC,</td>
</tr>
<tr>
<td>AFFECTED_ENTITY_NAME</td>
<td>The affected entity name</td>
</tr>
<tr>
<td>AFFECTED_ENTITY_ID</td>
<td>The affected entity ID</td>
</tr>
<tr>
<td>NUMBER_OF_EVALUATION_ENTITIES</td>
<td>Number of entities—Business Transactions, Applications, Tiers—violating the health rule conditions</td>
</tr>
<tr>
<td>EVALUATION_ENTITY_TYPE</td>
<td>APPLICATION, APPLICATION_COMPONENT (aka Tier), BUSINESS_TRANSACTION, APPLICATION_DIAGNOSTIC,</td>
</tr>
<tr>
<td>EVALUATION_ENTITY_NAME</td>
<td>The evaluation entity name (for JMX it is the counter name)</td>
</tr>
<tr>
<td>EVALUATION_ENTITY_ID</td>
<td>The evaluation entity ID, or &lt;NULL&gt; for JMX</td>
</tr>
<tr>
<td>NUMBER_OF_TRIGGEREDCONDITIONS_PER_EVALUATION_ENTITY</td>
<td>Number of times to loop through the triggered condition parameters. If one condition is triggered, the parameters repeat for each triggered condition.</td>
</tr>
<tr>
<td>SCOPE_TYPE_x</td>
<td>The scope of the parameter, whether the scope is the application, component, tier, node.</td>
</tr>
<tr>
<td>SCOPE_NAME_x</td>
<td>The name of the scope, such as ACME Book Store Application</td>
</tr>
<tr>
<td>SCOPE_ID_x</td>
<td>The scope ID</td>
</tr>
<tr>
<td>CONDITION_NAME_x</td>
<td>The health rule condition name</td>
</tr>
<tr>
<td>CONDITION_ID_x</td>
<td>The health rule condition ID</td>
</tr>
<tr>
<td>OPERATOR_x</td>
<td>Allowed operators: LESS_THAN, LESS_THAN_EQUALS, GREATER_THAN, GREATER_THAN_EQUALS, EQUALS, NOT_EQUALS</td>
</tr>
<tr>
<td>CONDITION_UNIT_TYPE_x</td>
<td>The condition for the threshold parameter: ABSOLUTE, BASELINE_PERCENTAGE, BASELINE_PERCENTILE</td>
</tr>
<tr>
<td>USE_DEFAULT_BASELINE_x</td>
<td>A Boolean parameter (true or false) applies only when the condition is equal.</td>
</tr>
<tr>
<td>BASELINE_NAME_x</td>
<td>Applicable only when the condition unit type is one of the baseline is true</td>
</tr>
<tr>
<td>BASELINE_ID_x</td>
<td>Applicable only when the condition unit type is one of the baseline is true</td>
</tr>
<tr>
<td>THRESHOLD_VALUE_x</td>
<td>Health rule threshold setting</td>
</tr>
<tr>
<td>OBSERVED_VALUE_x</td>
<td>Value that violated the health rule threshold</td>
</tr>
<tr>
<td>SUMMARY_MESSAGE</td>
<td>Summary of the notification, such as Health rules have been violated.</td>
</tr>
<tr>
<td>INCIDENT_ID</td>
<td>The incident identifier number for this health rule violation. In some cases it is defined as int(11) which means it takes four bytes of space, i.e., 2147483647 max value and -2147483648 min value. One bit is for the sign.</td>
</tr>
<tr>
<td>DEEP_LINK_URL</td>
<td>Controller deep link URL, such as: http://&lt;controller-host-url&gt;/#location=APP_II. Append the incident ID to the URL to provide a link to the Controller UI.</td>
</tr>
</tbody>
</table>
Parameters passed by an event

The parameters describe the event that triggered the action.

The total number of elements in the array depends on the number of event types and event summaries that triggered the action.

The parameter values are passed in the order in which they are described below.

**Structure of Parameters Sent by an Event**

- **APP_NAME**
- **APP_ID**
- **EN_TIME**
- **PRIORITY**
- **SEVERITY**
- **EN_NAME**
- **EN_ID**
- **EN_INTERVAL_IN_MINUTES**
- **NUMBER_OF_EVENT_TYPES**

The following parameters are passed for each event type:

- **EVENT_TYPE_x**
- **EVENT_TYPE_NUM_x**

- **NUMBER_OF_EVENT_SUMMARIES**

The following parameters are passed for each event summary:

- **EVENT_SUMMARY_ID_x**
- **EVENT_SUMMARY_TYPE_x**
- **EVENT_SUMMARY_SEVERITY_x**
- **EVENT_SUMMARY_STRING_x**
- **DEEP_LINK_URL**
- **ACCOUNT_NAME**
- **ACCOUNT_ID**

**Definitions of Parameters Sent by an Event**

<table>
<thead>
<tr>
<th>Event Notification Parameter</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>APP_NAME</td>
<td>Name of the business application</td>
</tr>
<tr>
<td>APP_ID</td>
<td>Application ID number</td>
</tr>
<tr>
<td>EN_TIME</td>
<td>Event notification time, for example, Wed Jan 04 09:36:55 PST 2012</td>
</tr>
<tr>
<td>PRIORITY</td>
<td>Integer designating how urgently a health rule violation should be fixed, with the lowest number (0) the most urgent</td>
</tr>
<tr>
<td>SEVERITY</td>
<td>Allowed values: INFO, WARN, or ERROR. In the AppDynamics UI they are called Info, Warning, and Critical</td>
</tr>
<tr>
<td>EN_NAME</td>
<td>Name of the event notification</td>
</tr>
<tr>
<td>EN_ID</td>
<td>Event notification ID number</td>
</tr>
<tr>
<td>EN_INTERVAL_IN_MINUTES</td>
<td>Event notification interval in minutes</td>
</tr>
<tr>
<td>NUMBER_OF_EVENT_TYPES</td>
<td>Determines how many times to loop through the event type map parameters</td>
</tr>
<tr>
<td>EVENT_TYPE_x</td>
<td>If there is more than one event type, the parameters repeat for each event type, where x increments the number representing the event type</td>
</tr>
<tr>
<td>EVENT_TYPE_NUM_x</td>
<td>Number of events of this type</td>
</tr>
<tr>
<td>NUMBER_OF_EVENT_SUMMARIES</td>
<td>Number of event summaries in the notification that determines how many times to loop through the event summary parameters</td>
</tr>
<tr>
<td>EVENT_SUMMARY_ID_x</td>
<td>Event summary ID number</td>
</tr>
<tr>
<td>EVENT_SUMMARY_TIME_x</td>
<td>Event summary time, for example: Wed Jan 04 09:34:13 PST 2012</td>
</tr>
<tr>
<td>EVENT_SUMMARY_TYPE_x</td>
<td>Type of event, such as: APPLICATION_CONFIG_CHANGE, APP_SERVER_RESTART, DIAGNOSTIC_SESSION, STALL</td>
</tr>
<tr>
<td>EVENT_SUMMARY_SEVERITY_x</td>
<td>Event severity, such as: INFO, WARN, or ERROR. In the AppDynamics UI they are called Info, Warning, and Critical.</td>
</tr>
<tr>
<td>EVENT_SUMMARY_STRING_x</td>
<td>Event summary string, such as: Application Server environment variables changed</td>
</tr>
<tr>
<td>DEEP_LINK_URL</td>
<td>http://&lt;controller-host-url&gt;/#location=APP_EVENT_VIEWER_MODAL&amp;eventSummary=Append each event summary ID to the URL to provide a link to the Controller UI for this event</td>
</tr>
<tr>
<td>ACCOUNT_NAME</td>
<td>Name of the account in which the action was triggered</td>
</tr>
<tr>
<td>ACCOUNT_ID</td>
<td>ID of the account in which the action was triggered</td>
</tr>
</tbody>
</table>

**Sample Custom Action Script**

See the [CreateServiceNow script](#), for an example of a script that creates ServiceNow tickets triggered by AppDynamics health rule violations.
Action Suppression

You can temporarily suppress a policy's automatic invocation of actions and alerts. You may want to do this while you are performing maintenance on or troubleshooting a component.

To see action suppression configurations created for an application, click Alert & Respond > Actions and then the Action Suppression tab. The list of action suppression configurations displays in the left panel with the objects affected by a selected configuration in the right panel.

The Object Scope list shows both tiers and nodes. Because the nodes are children of the tiers, action suppression affects them as well. Selecting an Application as Object Scope can produce a list of many items if the application is complex enough. All tiers and nodes are shown, although no business transactions are shown in the list.

Objects Affected by Action Suppression

You configure action suppression for a specific time period to apply to a specific object or several objects. The following entities can be the objects of action suppression:
- Application
- Business Transaction
- Tier
- Node
- JMX
- Machine

Within the time period configured for the action suppression, no policy actions are fired for health rule violation events that occur on the specified object(s).

You can also optionally disable reporting of metrics for an object for which actions are suppressed. Using this option can cause reported metrics for those objects to change without notice. If you see a sudden unexpected change in reported metrics for an object, check the action suppression configurations list to see whether action suppression with reporting disabled is currently active for that object.

If the object scope of an action suppression is at the node level, the suppression affects only node health rules. If the object scope of an action suppression is at the tier level, the suppression affects individual node health rules as well as tier-level health rules.

For example, if a tier-level health rule is configured to fire an action when a percentage of the nodes violates the condition, and then action suppression is configured on certain nodes in that tier, those nodes are still evaluated by the tier-level health rule.

Application-Level Action Suppression

In an application-level configuration, all entities in the application are affected.

Business Transaction Action Suppression

In a business-transaction-level configuration, you can suppress actions in:
- All business transactions in the application
- All business transactions within specific tiers
- Specific business transactions
- Business transactions with names having patterns that match certain criteria (such as all business transactions with names that start with "XYZ")
**Tier-Level Action Suppression**

In a tier-level configuration, all the nodes in the specified tier(s) are affected. You can suppress actions for:

- All tiers in the application
- Specific tiers

**Node-Level Action Suppression**

In a node-level configuration, you can specify the types of nodes for which to suppress actions:

- All nodes
- Java nodes
- .NET nodes
- PHP nodes

and within those types you can suppress actions for:

- All nodes
- Nodes in specific tiers
- Specific nodes
- Nodes with names, meta-data, environment variables or JVM system environment properties with matching criteria that you specify

**JMX-Level Suppression**

You can suppress actions on specific JMX objects, specific JMX instance names, and specific nodes.

**Machine-Level Action Suppression**

You can suppress actions run on specific machines. Actions run on all the nodes on the specified machine(s) are suppressed.

**Health Rules Affected by Action Suppression**

By default, an action suppression configuration applies to actions triggered by all events that are generated by the configured objects.

You can refine the configuration to apply only to actions triggered by specific health rule violations. For example, if an application contains HealthRuleA, HealthRuleB and HealthRuleC, but only HealthRuleC is configured for action suppression, actions will continue to fire for violations of HealthRuleA and HealthRuleB during the configured time period.

**Configure Action Suppression**

To access action suppression configuration, click **Alert & Respond > Actions**, then the Action Suppression tab.

You can edit, delete or create an action suppression from the Action Suppression Wizard that appears.

**Structure of the Action Suppression Wizard**

The Action Suppression Wizard contains three panels:

1. Overview panel sets the following:
   - Name: Action suppression configuration name
   - Disable Agent Reporting: Disable metrics reporting by agents associated with the affected objects during the action suppression
   - Start Time: Start action suppression
   - End Time: End action suppression

2. Object Scope panel sets the objects affected by the configuration.
   The options presented vary according to the scope defined.

3. Health Rule Scope panel sets the health rules that trigger action suppression. You can restrict action suppression to apply only to violations of specific health rules.

You can navigate among these panels by clicking their entries in the wizard. When you create a new configuration, configure the panels in order because the configuration of the scope determines the available affected objects presented in the object scope panel.
Disabling Health Rule Evaluation

Suppressing an action does not suppress the evaluation of any health rules linked to the action via a policy. Health rule violation events continue to be raised and surfaced to the UI when actions are suppressed.

**To turn off the evaluation of a specific health rule:**

Click **Alert & Respond > Health Rules**, select the Health Rule, click the edit option, and clear the **Enabled** checkbox in the Edit Health Rule Overview panel.

**To turn off all health rule evaluation for an application, user experience browser app, database, server, or analytics:**

Select the type of entity for which to disable health rule evaluation. Then clear the **Evaluate Health Rules** checkbox at the top left of the Health Rules window.
Predefined Templating Variables

On this page:
- Forming Variable Names
- Base Names
- Info Class Fields
- Examples

Related pages:
- HTTP Request Actions and Templates
- Email Templates

This topic describes the predefined variables that can be used in an HTTP request template or in an Email template.

The policy engine substitutes the value of the variable in the context of the triggering event(s) when it sends the actual request or email.

Forming Variable Names

To form a predefined variable name, combine the base name of the variable with a field of its corresponding info class. You can use the base name with the corresponding info class only for example, you can use the base name `action` with info class `ActionInfo` only. If an info class has no fields associated, use the base name as the variable. For example, to use the controller URL as a variable in a template, use `{controllerURL}`.

If an info class comprises multiple fields, select a single field and associate it with the base name. For a list of fields associated with an info class, refer Info Class Fields.

For example:

- to use the account name as a variable in a template, combine the base name `account` with the `name` field of the `EntityInfo` class to form the variable `${account.name}`.
- to use the trigger time of an action as a variable, combine the base name `action` with the `triggerTime` field of the `ActionInfo` class to form the variable `${action.triggerTime}`.

You can chain the segments of a variable name where the info class field type is yet another info class.

For example, to use the name of an application in which the latest triggering event occurred:

1. Combine the `latestEvent` base name with the `application` field of the `EventInfo` class.
2. The field type of the `application` field is `EntityInfo` which is yet another info class, hence select the name field of the `EntityInfo` class.
3. Form the variable name by appending the segments:

   `{<base name>.<field name of info class1>.<field name of info class2>}`

   `$.latestEvent.application.name`.

Base Names

These are the base names for the predefined variables with their corresponding info classes:

<table>
<thead>
<tr>
<th>Base Name</th>
<th>Description</th>
<th>Info Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>account</td>
<td>Account in which the action was triggered</td>
<td><code>EntityInfo</code></td>
</tr>
<tr>
<td>policy</td>
<td>Policy that triggered the action</td>
<td><code>PolicyInfo</code></td>
</tr>
<tr>
<td>action</td>
<td>Triggered action</td>
<td><code>ActionInfo</code></td>
</tr>
<tr>
<td>topSeverity</td>
<td>INFO, WARN or ERROR</td>
<td><code>NotificationSeverity</code></td>
</tr>
<tr>
<td>topSeverityImage</td>
<td>Severity image or icon</td>
<td><code>ImageInfo</code></td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
<td>Type</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>notificationConfigText</td>
<td>Email actions only</td>
<td>String</td>
</tr>
<tr>
<td>controllerUrl</td>
<td>URL of the controller</td>
<td>String</td>
</tr>
<tr>
<td>appDynamicsIcon</td>
<td></td>
<td>ImageInfo</td>
</tr>
<tr>
<td>appDynamicsLogo</td>
<td></td>
<td>ImageInfo</td>
</tr>
<tr>
<td>latestEvent</td>
<td>Most recent triggering event</td>
<td>EventInfo</td>
</tr>
<tr>
<td>fullEventList</td>
<td>List of events that triggered the action</td>
<td>List&lt;EventInfo&gt;</td>
</tr>
<tr>
<td>fullEventsByTypeMap</td>
<td>List of events that triggered the action, grouped by type</td>
<td>Map&lt;String, List&lt;EventInfo&gt;&gt;</td>
</tr>
<tr>
<td>clampLimit</td>
<td>Optional setting, limit on how many triggering events to display</td>
<td>int</td>
</tr>
<tr>
<td>clamped</td>
<td>True if clamp limit is set</td>
<td>boolean</td>
</tr>
<tr>
<td>clampedEventList</td>
<td>( n ) most recent triggering events if clamped</td>
<td>List&lt;EventInfo&gt;</td>
</tr>
<tr>
<td>clampedEventsByTypeMap</td>
<td>( n ) most recent triggering events if clamped, grouped by type</td>
<td>Map&lt;String, List&lt;EventInfo&gt;&gt;</td>
</tr>
<tr>
<td>fullEventsNodeMap</td>
<td>Node level variables to be used in email and HTTP templates</td>
<td>Map&lt;EventInfo, NodeTemplateVariables&gt;</td>
</tr>
</tbody>
</table>

**Info Class Fields**

Use the fields in the appropriate class, based on the table above, to form the variable to use in a template.

```java
class EventInfo {
    EventType eventType
    long id
    String guid
    String eventTypeKey
    Date eventTime
    String displayName
    String summaryMessage
    String eventMessage
    EntityInfo application
    EntityInfo tier
    EntityInfo node
    List<EntityInfo> affectedEntities
    boolean healthRuleEvent
    EntityInfo healthRule // * Only defined when healthRuleEvent == true
    EntityInfo incident // * Only defined when healthRuleEvent == true
    boolean healthRuleViolationEvent
    NotificationSeverity severity
    ImageInfo severityImage
    boolean btPerformanceEvent // * true when eventType matches one of the BT performance event types
    String deepLink
}
```
class ImageInfo {
    String name
    String fileName
    String mimeContentRef
    String deepLink
}

class EntityInfo {
    EntityType entityType
    String entityTypeDisplayName
    long id
    String name
}

class PolicyInfo {
    EntityType entityType
    String entityTypeDisplayName
    long id
    String name
    boolean digest
    digestDurationInMins
}

class ActionInfo {
    EntityType entityType
    String entityTypeDisplayName
    long id
    String name
    Date triggerTime
}
enum NotificationSeverity { INFO, WARN, ERROR }

class NodeTemplateVariables {
    private long id;
    private String name;
    private long tierId;
    private String tierName;
    private long machineId;
    private String machineName;
    private boolean machineAgentPresent;
    private String machineAgentVersion;
    private boolean appAgentPresent;
    private String appAgentVersion;
private String ipAddresses;
private AgentType agentType;
}

Examples

HTTP request example for most recent triggering event

```
http://myController:8080/controller/rest/applications/${latestEvent.application.name}/nodes/${latestEvent.node.name}
```

HTTP request example iterating through a list of triggering events

```
#foreach(${event} in ${fullEventList})
http://myController:8080/controller/rest/applications/${event.application.name}/nodes/${event.node.name}
#end
```

Email with dynamically varying subject example

If you have several health rules to monitor your application and one policy to send email notification for all the health rules, you might find it difficult to search for the right information in the notification. You can define a variable to dynamically update the email subject based on the event that triggered the notification.

Ensure that you define the subject variable in the email body and include it in the subject field.

```
Subject: $subject
Body:
...
<!-- Subject changes dynamically based on the triggering event -->
#set($subject = #foreach(${events} in ${fullEventList})
${events.eventType} #end )
...
```

Email HTML body example

AppDynamics uses the following email template to notify the details of the events occurring on an application. You can customize the required event details you want notified by enabling the event types.
Event Notification for the "${latestEvent.application.name}" Application
Event Notification Name: ${policy.name}
Event Notification Severity: ${topSeverity}

Summary of events occurring during the
${policy.digestDurationInMins}+ minute(s) prior to
${action.triggerTime}:

## Summary table
|| Event Type | Count ||
#foreach($eventTypeEntry in $fullEventsByTypeMap.entrySet())
|| ${eventTypeEntry.getKey()} | ${eventTypeEntry.getValue().size()} |
#end

## Full List of Events
The following events occurred during the time frame:
#if ($clamped)
Warning: The event list has been clamped at ${clampLimit} results!
Please see the event and/or request snapshot viewers for the full list of events.
#end

|| Event Time | Event Type | Severity | Tier | Node | Summary ||
#foreach($eventList in $clampedEventsByTypeMap.values())
#foreach($event in $eventList)
|| ${event.eventTime} | ${event.displayName} | ${event.severity} |
$!{event.db.name} | $!{event.tier.name} | $!{event.node.name} | ${event.summaryMessage} |
#end
#end

Event Notification from AppDynamics.
This is an auto-generated email summarizing events on the
"${latestEvent.application.name}" application. You are receiving this because you are configured as a recipient on the "${policy.name}" event notification. This is not necessarily an exhaustive list of all events during this time frame. Only those event types enabled in the notification will appear in this message.
Example to extract event details using iteration

The following examples iterates through the eventList to extract the details of each event occurring during the specified time.

<h1>Summary of events occurring during the
${policy.digestDurationInMins}+ minute(s) prior to
${action.triggerTime}:</h1>
<table>
#foreach(${eventList} in ${fullEventsByTypeMap.values()})
#foreach(${event} in ${eventList})
<tr>
<td>
<!-- Event icon -->
<img src="${event.severityImage.mimeContentRef}"
alt="${event.severity}" />
</td>
<td>
<!-- Event name with event link -->
<a href="${event.deepLink}">${event.displayName}</a>
</td>
<td>
<!-- Event message -->
${event.eventMessage}
</td>
</tr>
#end
#end
</table>

Example code to list all variables

Copyright © AppDynamics 2012-2019

Page 326


variables below

${fullEventsNodeMap}

map elements below

#foreach(${value} in ${fullEventsNodeMap.values()})
${value}
${value.name}
#end

If these predefined variables do not meet all your needs, you can define your own custom variables for use in these templates in the Create Template window.

Do not use any of the predefined variable names for your custom variables.
Email Digests

In addition to specific actions that are triggered by specific events, you can create an email digest that reports a summary of specific events to a recipient list on a schedule.

Permissions

To create, edit, or delete email digests, a user must have the Configure Actions, Configure Health Rules, and Configure Policies permissions.

Sample Email Digest

This is a sample of an email digest that is sent every hour:
Create an Email Digest

To create an email digest:

Click Alert & Respond in the menu bar, then Email Digests either in the right panel or the left navigation pane. Follow the steps in the wizard to create the digest.
Dashboards and Reports

Dashboards and Reports are useful for getting a high-level overview of the health and performance of your system on a regular basis.

Dashboards provide a graphical overview of the selected data made available for quick access. Custom dashboards let you create and arrange widgets to give users a visual overview of the data of interest to them. A War Room is similar to a dashboard, but it is interactive. The chat facility in virtual war rooms enables visually-enhanced collaborative troubleshooting in real time in response to high priority events.

Reports capture AppDynamics data from dashboards and help share it with the recipients. Reports can be automatically generated on a regular schedule.

- Virtual War Rooms
- Widgets
- Custom Dashboards
- Reports
Virtual War Rooms

A war room is a combination of interactive dashboard and notes. It integrates a custom dashboard being created in real-time combined with collaborative troubleshooting in the form of a focused chat room. While troubleshooting an issue, you can add notes and data that are visible to all the war room participants immediately in a timeline format.

The virtual war room helps to:

- Get everyone on the same page by looking at the same real-time data
- Keep the focus on metrics that translate to the business value the application delivers
- Foster communication among a broad audience
- Identify resolution criteria and assign ownership for resolution tasks

Start a War Room

Start a virtual war room, a collaborative chat room, when you want to troubleshoot an issue on-the-go.

You must have the account-level permission, Create War Rooms to start a War Room.

1. Start creating a war room using one of the following three options:
   - From Applications > Troubleshoot > War Rooms > My Active War Rooms, click Start a War Room.
   - From Applications > Application Dashboard > Events > Actions, click Start a War Room. Access the events list through the Events tab available in many of the built-in dashboards.
   - On the Jira issue page, from the AppDynamics Incident panel, click Start or Join a War Room associated with this Incident. See AppDynamics Add-on for JIRA.

2. Name your war room and click OK if you want to create a new room or select From Template to create a new room using available war room templates. See War Room Templates.
3. Click Add Widget to add your widgets to the war room. See Widgets.
4. Click Save.

You can view all the war rooms you have created under My Active War Rooms tab and edit your widgets anytime through Actions menu.

**War Room Templates**

You can re-use the layout of an existing war room by saving a war room as a war room template. This can save time in creating new war rooms with numerous widgets.

Save an existing war room as a template while working on it or just before closing it.

**To create a war room template:**

1. In the Actions menu of the current war room using which you want to create a template, click Save as Template.
2. Provide a name for the template and an optional description.
3. Click OK.

The newly created template is now available for reuse in creating other war rooms. You can view all your war room templates under War Rooms > Templates.

**To delete a war room template:**

1. Click War Rooms > Templates.
2. From the list of war room templates, select the template that you want to delete.
3. Click Delete Template and confirm the deletion.

**Manage Presenters**

The user who starts a war room is the first presenter.

Only the presenter can:

- Modify the collaboration space to add, delete, and configure widgets
- Make any other participant a presenter. There can only be one presenter at a time
- Share and stop sharing the war room with other participants
- End the war room

Presenters can also do everything that participants can do.

**To change presenter:**

Right-click on any participant name in the participant list and select Make Presenter.

The ability to edit and control sharing of the collaborative dashboard and to end the war room passes to the new presenter. Other participants cannot see the war room until the presenter shares it. This gives the presenter the opportunity to set up the widgets privately before showing the war room to others.

**To end the war room:**

Click the End War Room button. When prompted, confirm that you want to end it.

Only the presenter can end the war room. A war room is automatically ended after 60 minutes of inactivity. Inactivity means no participants are viewing the war room window.

**Manage Participants**

Everyone who has joined the war room but who is not currently a presenter is a participant. A participant can be:

- An AppDynamics participant is an authenticated UI user with access to the AppDynamics account that is hosting the war room.
- A guest participant does not log into the AppDynamics UI, but has been invited to join by an AppDynamics participant. This type of user functions as an anonymous, read-only participant.
AppDynamics participants can:

- View a war room to which they have been invited
- Invite other participants
- Add notes
- Start a new war room, if the participant has Create War Rooms permission

Guest participants can view a war room to which they have been invited.

**To view a war room:**

You can view only the war rooms in which you are a presenter or a participant. The list does not display all the war rooms currently in progress on the controller.

To view a war room, you should have created one, got invited to one, or have access to the shared URL available to anonymous users. Perform one of the following to view a war room:

1. Click the link in the invitation that you received to join the war room.
2. Click Troubleshoot > War Rooms, then the war room that you want to join from the war rooms list.

**To invite another participant to the war room:**

Send a link to invite another participant to the war room as follows:

1. From your active war room, click + | Participants.
2. From the Invite Users widow, copy the URL for joining this War Room.
3. Email or SMS the link to the persons whom you are inviting, with instructions to click the link in a browser window.
4. Click OK to close the Invite Users window.

After clicking the link, the recipient is invited to log in as an AppDynamics participant or as an anonymous guest.

**To add a note:**

1. From your active war room, click + | War Room Notes.
2. On the Enter Notes window, do the following:
   a. Enter your note in the text field that appears.
   b. (Optional) Qualify your note as good or bad.
   c. Click Post.

**Automatic Save, Synchronize, and Archive a War Room**

The war room is automatically saved and all participants' views of it are synchronized whenever the presenter makes a modification to the dashboard. The saving and synchronizing happens continually and is transparent to the participants when it occurs. In addition, the presenter can always force a save and sync manually by clicking the Save icon at the top of the war room dashboard.

If the controller is restarted while a war room is in-progress, the war room is saved and can be accessed when the controller comes back up.
Widgets

On this page:

- Widget Types
- Widget Properties
- Configuring Widgets

Related pages:

- Create and Manage Custom Dashboards and Templates
- Using Wildcards in Metric Definitions
- Configuration Import and Export API
- Visualize Analytics Data

You can use widgets to create visual representations of your data in custom dashboards and war rooms.

This page describes the widgets available with Application Performance Monitoring. If your platform is licensed for Analytics, you can display the results of Analytics searches on custom dashboards as well. See Visualize Analytics Data for more information.

Widget Types

Most of the widgets types available in custom dashboards and war rooms will be familiar to you if you have worked with numeric charts before. This section provides information for certain widget features that may require additional explanation.

Time Series Graph

The standard time series graph displays one or more values over time in various formats: line, area, column, scatter diagram.

If you configure a widget for a war room that examines metrics on a Java node, consider the streaming graph widget described in Virtual War Rooms. The streaming widget updates more frequently (every 5 seconds) than the standard time series widget (every 60 seconds) for collaborative real-time troubleshooting.

Events are grouped by time of resolution at the bottom of the graph. You can choose what events to display when you enable the Events Overlay and configure the event queries for the graph.

Pie Chart

Pie charts or doughnut charts display multiple values as proportional slices of a whole. The data series should meaningfully total 100% of some series. For example, CPU Busy + CPU Idle + CPU Stolen on a single node or tier.

Gauge Widget

The gauge widget displays the magnitude of a metric value with respect to minimum and maximum values, both by the extent to which the gauge is filled and by the color of the fill.

You can set the minimum and maximum values displayed by the widget. Set the maximum to the highest acceptable maximum or a value close to it so that the gauge is configured for metrics within acceptable values for your site. If you do not set them, default values are used.

The following table explains the color coding for gauge widgets:

<table>
<thead>
<tr>
<th>Percentage</th>
<th>Color</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-10%</td>
<td>Green</td>
<td>OK</td>
</tr>
<tr>
<td>10-50%</td>
<td>Gradient from green to yellow</td>
<td>OK trending to WARNING</td>
</tr>
<tr>
<td>50-90%</td>
<td>Gradient from yellow to red</td>
<td>WARNING trending to CRITICAL</td>
</tr>
<tr>
<td>90-100%</td>
<td>Red</td>
<td>CRITICAL</td>
</tr>
</tbody>
</table>
For example, the following gauge displays the calls per minute metric without any special configuration for minimum and maximum values. The current value is 74 cpm, which is midway between the minimum, 0 and maximum, 148 for the time period, in the WARNING range.

![Gauge Example](image)

If you set the max value to 1000, the gauge color changes to green because 74 falls within the OK range.

**Streaming Graph Widget**

A streaming graph widget is similar visually to a time series graph, but it is updated every five seconds in contrast to every 60 seconds in the time series graph.

For Java nodes only, war room supports the streaming graph widget. It displays data from only a single specified node and only a Java node. This feature is not supported for any of the other agents. To display streaming data from multiple nodes you need to create multiple widgets.

**Log Tail Widget**

The war room supports a log tail widget used for Java nodes only to share a log file or parts of one.

The log tail widget displays the contents of the file specified by its Path property. The Path is the absolute path to the log file on the machine on which the selected node is running. If you provide a regular expression in the widget properties, the widget displays only those lines in the log file that match the regular expression. If no regular expression is provided, the widget displays the entire log file. Note that applying a regular expression filter will only adjust lines that come after you have specified the filter. Previous lines will remain filtered based on the previous rule. The widget uses the Java conventions for regular expressions.

If the log file rolls over during the request, the response to the log tail request is undefined.

**Metric Value**

You can specify a title for the widget as well as a format string that adds the name of the selected metric, the time range, or the name of the function. Alternately, you can use a separate label widget to explain the metric value.

You have the option of formatting metric values by abbreviating them, setting a decimal limit, and removing trailing zeros. Abbreviations only apply to numbers over one million. The ability to set the number of decimal places allows you to display decimal values less than one, as well as decimals for larger numbers. If your decimal limit is set to zero, then the remove trailing zeros feature is turned off by default.

**Health Status**

This is a health bar, filtered table, or status light circle of health statuses for entities, such as applications, business transactions, tiers, nodes, databases, servers, and health rules. After choosing an entity type to show the health for, you have the option to narrow the scope or simply display all.

It is important to note that when selecting tiers as your entity, the color of the health status depends on the statuses of all the health rules that are configured for the tiers. For example, if there are four health rules for the tiers, then the color of the widget will reflect the
cumulation of all of their statuses. The widget displays green for normal, yellow for warning, and red for critical status. See Configure Health Rule Conditions in Configure Health Rules for information on how the metrics that define warning and critical health status are set.

When using the circle visual, you have the option of manipulating the display by choosing its aggregation type: either Ratio Graph or Most Severe Status.

Event List

This is a filtered table of events that occurred within the specified time range.

IFrame

You may want to include a shared custom dashboard in another custom dashboard when you have different operators who focus on particular dashboards and you want to roll them all up into a master dashboard for management.

To display information from another dashboard or from another website inside this custom dashboard, you need the URL. You can get the URL for a screen by clicking Settings > Copy a link to this screen to the clipboard.

The iframe widget is sandboxed. Advanced HTML features, such as forms and Javascript, will not work with the iframe widget.

Widget Data

Some widgets, such as a time series graph, can display a single value or multiple values. Some widgets, such as a doughnut, must display multiple values to make sense. Others, such as a gauge or a metric value, display only a single value.

Specifying a function of a metric value for a single entity

When you specify the data to be displayed in a widget, you use a pulldown menu in the Add Series window to select which function of the value to display:

- Minimum: Minimum value, only available for averaged metrics
- Maximum: Maximum value, only available for averaged metrics
- Value: Contains the average or the sum across the time range depending on the metric
- Sum: Aggregated value of the metric over the time range
- Count: Count of the observed values over the time range
- Current: Sum of the most recent minute’s metric data values across all the included nodes

You can also use a metric expression to specify a metric. The behavior is similar to using an expression for a health rule. For more information about how to use metric expressions, see how to build an expression for Health Rules.

Specifying metrics in multiple entities using a wildcard

You can use a wildcard to specify a metric that evaluates across several similar entities for metrics in the following branches in the metric hierarchy:

- Hardware Resource
- JVM
- CLR
- Custom Metrics

See Using Wildcards in Metric Definitions for instructions on how to do this.

Empty Data Series

It is possible that no data is available for a particular metric within the specified time range while for other time ranges valid data is reported. When this occurs, the label for the empty data series appears in the legend but no data is displayed. For example, no line in a time series graph or slice in a pie chart.
Enabling limits on the number of JMX instances or nodes for a single entity

You can use account level configurations to enable dashboard or widget limits on large numbers of JMX instances or nodes instrumented in one application while returning widget data. This also includes threshold checks for business transactions, errors, servers, and backend data. This applies to the JMX type metric in the widget Type of Nodes menu. The All Nodes in the Application option becomes disabled if the number of nodes exceeds the threshold. An error is returned if the number of any threshold is exceeded on the backend. This should help prevent outages.

Widget Properties

Widget properties vary to some extent based on the type of the widget.

Most of the widget properties in the configuration forms are self-explanatory. The following list describes some of the less obvious properties:

- Metric Display Name: You can choose the display name for the metrics in a widget. Specify one or more of the following substitution variables to determine the metric display name:
  - ${#}: Number the displayed metrics based on sorting with the rollup value.
  - ${m}: Metric name.
  - ${e}: Entity name.
  
  For example, if specify ${m}, the widget displays the full metric name.

- Static Thresholds: You can create static threshold with a specific value for graph widgets. The threshold appears as a line on the graph.

- Double click action: Some widgets have an option to define the result of double-clicking the widget. You can configure the double click action to open the Metric Browser or a URL specified by the Drilldown URL value.

- Drilldown URL: The Drilldown URL specifies the URL of an AppDynamics window you want the widget to open when you double-click the widget. For example, if you see an increase in slow requests and want to investigate the cause, you may want to jump to the App Servers List of all tiers from your dashboard by double-clicking a widget. To get the URL for a specific screen, click Settings > Copy a link to this screen to the clipboard.

- Time Range: Widgets can use the global or dashboard time range or a time range that is specific to the widget. If you choose a widget-specific time range, select the widget time range from the pulldown menu.

- Events Overlay: You can configure specific events to overlay on the graph widget to show how specific events may correlate with performance. Check the Show Events check box and create the query to define which events should be overlaid.

Configuring Widgets

Set up or modify widgets with the widget palette in edit mode.

If you configure widgets for a war room, you will see an additional category for Streaming Data widgets. To add widgets to the war room space the presenter must be in edit mode. To enter edit mode, click edit toggle button which is always visible. If it is not visible and you are the presenter, you are already in edit mode. You can duplicate one or more widgets at a time and delete any widget as required.

To configure a new widget:

1. Click Edit.
2. Click Add Widget. The widget palette appears.
3. Click the widget in the palette. The configuration form appears on the left side of the configuration window.
4. Configure both the widget data and the widget properties. After you configure the data, a preview of the widget appears. The preview updates as you set more properties.
5. Click Save.

To modify an existing widget:

1. Click Edit.
2. Click the widget in the dashboard.
3. Click the settings icon in the upper right corner of the widget.
4. Use the form to modify the widget.
5. Click Save.
Using Wildcards in Metric Definitions

On this page:

- Define the Relative Metric Path
- Modify the Relative Metric Path
- Configure the Metric for Multiple Entities
- Verify the Metric Specification
- Wildcards Replace Entire Path Segments
- Handling Colons in the Metric Path

Related pages:

- Configure Health Rules
- Create and Manage Custom Dashboards and Templates
- Metric Browser

In a custom dashboard widget or health rule, you can use a wildcard to specify a metric that evaluates across several entities, such as multiple hardware entities or memory pools.

The wildcard feature is supported for metrics in the Hardware Resources, JVM, and CLR branches of the metric hierarchy and for custom metrics created with a monitoring extension as described in Extensions and Custom Metrics. To use wildcards, do the following:

1. **Define the relative metric path** to the metric that you want to display. This involves getting the full metric path in the Metric Browser and then trimming it.
2. **Modify the relative metric path** to apply to multiple entities.
3. **Paste the modified relative path** in the metric selector for the widget or health rule that you are configuring.

**Define the Relative Metric Path**

A metric path is a pipe-delineated path to a specific metric. In the AppDynamics UI, hover over a metric in the metric browser to get the full metric path. Right-click on the metric to view Copy Full Path option.

To define the relative metric path, truncate the leftmost part of the full metric path.

To know how much to truncate for a particular use case, look at the embedded metric browser in the Metric Selection window of the widget or health rule configuration that you are setting up. By the time you have reached this selector, you will have configured the application, tiers or nodes of the metric that you want to select.

Use the category in the Metric Selection window as the first segment of the relative metric path, truncating everything from the full metric path that comes before that segment. In the example below the first segment would be Agent or Hardware Resources or JVM or CLR.

As per the preceding example, the full metric path copied from the Metric Browser is Application Infrastructure Performance|ECommerce Server|Hardware Resources|Disks|dev-dm-1|% CPU Time.
Truncate everything to the left of the category selected in the Metric Selection Window that is, Hardware Resources in this example, as the display in the embedded metric browser starts with Hardware Resources.

The relative metric path is Hardware Resources|Disks|dev-dm-1|% CPU Time.

Consider another example where the full metric path copied from the metric browser is, Application Infrastructure Performance|ECommerce Server|Individual Nodes|ECommerceAppNode|Hardware Resources|Disks|dev-dm-1|% CPU Time.

The relative metric path after truncating everything to the left of the Hardware Resources is Hardware Resources|Disks|dev-dm-1|% CPU Time.

Modify the Relative Metric Path

Replace a single segment in the relative metric path with an asterisk to indicate that the metric should be evaluated for all the entities represented by that segment.

For example:

The relative metric path is, Hardware Resources|Disks|dev-dm-1|% CPU Time and you want to display or create a health rule condition on the % CPU time for all of the disks in that tier or node.

Substitute the asterisk for the disk name as Hardware Resources|Disks|*|% CPU Time.

Configure the Metric for Multiple Entities

In the Metric Selection window for the dashboard widget or health rule, you can configure metric for multiple entries.

To configure the metric for multiple entities:

Select Specify Relative Metric Path on the bottom of the Metric Selection window, paste the modified, wild-carded relative metric path and click Select Metric.

Verify the Metric Specification

When you configure a relative metric specification for multiple similar entities in a custom dashboard, you see the multiple metrics displayed in the widget.
Wildcards Replace Entire Path Segments

An asterisk replaces the entire segment in the path. It cannot be used to replace only a portion of a string, as they are used in regular expressions.

For example, the following usage in an attempt to get the ART for all the business transactions beginning with View is not valid and not supported:

Business Transaction Performance|Business Transactions|ECommerce Server|View*|Average Response Time (ms).
Specify, Average Response Time (ms) to get ART for all the business transactions on the ECommerce Server.

Handling Colons in the Metric Path

The colon, if any, in the metric path is translated into a pipe. Only one segment is allowed between pipes. It is to be noted that the colons and pipes can be used interchangeably.

For example:

The colon in the metric path, Application Infrastructure Performance|ECommerce Server|JVM|Memory|Heap|Committed (MB) is translated to Application Infrastructure Performance|ECommerce Server|JVM|Memory|Heap|Committed (MB).

For the relative metric path, to get metrics for both heap committed and non-heap committed, specify JVM|Memory[^]*|Committed (MB). Do not specify JVM[^]*|Committed (MB).
Custom Dashboards

A custom dashboard enables you to display a specific set of metrics and data points on one screen. You can display application, server, and database metrics reported by AppDynamics agents. You can also supplement the AppDynamics built-in dashboards—such as the application dashboard, tier dashboard, and so on—and display metrics that are of interest to a particular audience.

You can use custom dashboards to present selected metrics for a user who only needs a relatively narrow or focused view of the data. For example, an executive may only need a high-level view of system performance and activity. You can assign such users to the predefined Custom Dashboard Viewer role. Any user with a custom dashboard viewer role can view custom dashboards.

Instead of using the Custom Dashboard Viewer role, you can share a custom dashboard.

Use controller-level custom dashboards to:

- Provide a customized view of application, server, and database performance data.
- Aggregate data from different applications on the same Controller.
- Compare data from different applications on the same Controller.
- Show a single view of both live and historical data.
- Share data with other users and stakeholders.

**Create Custom Dashboards**

To create a Controller-level custom dashboard, click Dashboards & Reports > Dashboards > Create Dashboard. In the Dashboards panel, click an existing dashboard to edit it.

To create or edit a custom dashboard, a user needs to have the Can Create Custom Dashboards permission. Also, see Visibility and Permissions.

**Custom Dashboard Templates**

Custom dashboard templates are an efficient way to display the same type of information for multiple tiers or nodes. You create a dashboard in the scope of a tier or node and then use that dashboard as a template to display the same type of data for other tiers or nodes in the same application. Associating the template with a specific tier or node sets the source of the dashboard's data.

To create or edit a custom dashboard template, a user needs to have the Configure 'My Dashboards' for Tiers and Nodes permission.
• Compare overall application or infrastructure performance against individual tier or node performance
• Show a standard set of metrics for multiple tiers and nodes

You create and access dashboard templates from the My Dashboards tab of the built-in tier or node dashboard. From there you can create a new dashboard template for that tier/node or use dashboard templates that were created for other tiers or nodes in the same application by association.

Although the process for configuring Controller-level and tier or node-level dashboard templates is similar, the two sets of dashboards are separate. In other words, you cannot associate a Controller-level dashboard with a tier or node through the My Dashboards tab, nor can you access from the Controller-level dashboards list any tier and node level dashboards outside the application in which they were created.

The dashboard and template names must be unique controller-wide. You cannot create a custom dashboard using an existing dashboard template name, nor can you create a dashboard template using an existing custom dashboard name.

You can export custom dashboards and dashboard templates to a JSON file and then import them where you need them.

Viewing Strategies

For AppDynamics Users

If you have set up default view permissions to custom dashboards, any user who is logged into the Controller can view a custom dashboard.

Users can also be granted view, edit, and delete permissions on specific dashboards.

The Dashboards Viewer predefined role has permissions to view custom dashboards. This role can be granted to users or groups who need access to custom dashboards. See Custom Dashboard Permissions for other details.

For the Public

Sharing a controller-level custom dashboard makes the dashboard available to viewers who do not have AppDynamics accounts.

If the dashboard is shared, anyone who has been sent a direct link to the dashboard URL can view it. If someone has the URL and the dashboard is not shared or no longer shared, they see a message that the resource is not available when they try to access that URL.

To control Controller-level custom dashboard sharing, click Share on the menu of the custom dashboard configuration.

There is no comparable sharing mechanism for external viewers without controller credentials to access tier- and node-level custom dashboards based on templates. To access these types of dashboards the user must be logged into the controller and have custom dashboard view permissions on the specific dashboard.

Simplifying the Data Displayed

If a chart in a custom dashboard is displaying more data than you want to see, click the legend for the data that you do not want to see. This will remove that data from the chart. You can click the greyed-out legend to restore the display.
Create and Manage Custom Dashboards and Templates

In the Dashboards list, select the dashboard that you want to edit, delete, copy, share, export, or import and click the appropriate button.

---

This topic describes how to create and manage dashboards.

Permissions to create, view, edit and delete custom dashboards are set through roles. See Custom Dashboard Visibility and Permissions in Custom Dashboards as well as Roles and Permissions for general information about setting permissions.

Access and Create Custom Dashboards

For controller-level dashboards:

1. Click Dashboards & Reports in the menu bar to view the list of existing custom dashboards or to create a new one.
2. If there are existing dashboard names displayed, click Dashboards to see them.
3. Click + Create Dashboard to create a new custom dashboard.
4. Add widgets to the dashboard.
5. To save the dashboard, click OK in the Create Dashboard window.

For tier/node-level dashboard templates:

1. In an application, in the built-in tier or node dashboard in which you want to create or use a custom dashboard template, click the My Dashboards tab.
2. If there are no associated custom dashboards for the entity, click + Create a new Custom Dashboard to create a new one.
3. If you want to associate an existing dashboard with the entity, click Associate Dashboards from the dropdown list of the dashboard with which you want to associate the entity. See Associate a Custom Dashboard Template with a Tier or Node for more information.

A dashboard has no content until you add widgets to it. See Widgets for details.

General Dashboard Characteristics

- Name: Dashboard names must be unique per controller.
- Two layout types:
  - Grid: This is the default type that gives you a flexible layout that is easy to rearrange on the canvas. Grid layout also scales in size when viewed on mobile devices. You cannot overlap widgets using this layout type.
  - Absolute: Use this type to control width and height and exact placement of widgets on the canvas. With absolute layout, you can overlap widgets and use Send to Front or Send to Back actions to control what is displayed.
- Configurable auto-refresh Interval: Sets the data refresh rate in seconds.
- Dashboard Time Range: This setting applies only to controller-level dashboards. From the Time range dropdown list you can set a global time range for the dashboard. When you add widgets, you can specify whether the widgets use this global time range or a widget-specific time range. If you do not specify a widget-specific time range, the global time range is used for all the widgets.
Accessing the Dashboard Operations

While you are in edit mode, your changes are automatically saved. To save changes manually, disable Auto-Save and click the save icon. There is no way to save your changes manually. If you are accessing a dashboard on a mobile phone click the Mobile icon for a better viewing experience.

To access various operations concerning the dashboard, click Actions.

Widget Operations

You must be in edit mode to add and modify widgets. See Widgets for details on setting up widgets.

To access the widget palettes:

In the dashboard configuration window, click + Add Widget.

To add a widget to the dashboard:

1. In the left panel of the Add Widget window select the context.
2. Click the widget in the palette on the right.

To view and edit the widget properties window:

Click the gear icon in the upper right corner of the widget.

To delete one or more widgets:

Select the widgets and click Delete. Or right-click the widget and select Delete.

To duplicate one or more widgets:

Select the widgets and click Duplicate. Or right-click the widget and select Duplicate.

To copy/paste a widget:

Select the widget and click Copy, then click Paste.

or

• Select the widget, right-click, and select Copy.
• Right-click the widget again, and select Paste.

To edit the properties of multiple widgets:

1. Hold down your command key and select the widgets you want to edit the properties for.
2. Click Edit.
3. From the Edit <number of widgets> Widgets dialog box, select the properties you want to apply to all the widgets, and click OK.

Note that not all widgets share the same properties, and Analytics widgets cannot be edited using the multi-widget editing feature.

Delete or Modify a Custom Dashboard

For existing controller-level custom dashboards:

1. Click Dashboards & Reports.
2. Click Dashboards.
3. In the Dashboards list, select the dashboard that you want to edit, delete, copy, share, or export and click the appropriate button.

For existing dashboard templates:

In the My Dashboards tab of an entity associated with the dashboard:

• To modify an existing dashboard template, select it in the list and click Edit, pencil.
• To remove an existing dashboard, click Manage Dashboards from the drop-down list, select the template in the list, and then click Delete.

Associate a Custom Dashboard Template with a Tier or Node

The My Dashboards tab of the built-in tier and node dashboards provides access to custom dashboards that are associated with the tier or node.

To view an associated custom dashboard populated with data from the entity, click the dashboard in the list displayed by the My Dashboards tab.

To associate one or more existing dashboard templates with the entity, do the following:

1. Click Associate Existing Dashboard with the Tier/Node for an entity that has no associated dashboards. Alternatively, click Associate Dashboards from the dropdown list for an entity that already has at least one associated dashboard.
2. In the dashboard picker, move the dashboards to associate from the Available Dashboards list to the Selected Dashboards list.

To disassociate a dashboard template with an entity:

1. In the dropdown list, click Associate Dashboards.
2. In the dashboard picker, move the dashboards to disassociate from the Selected Dashboards list to the Available Dashboards list.

Dashboard Templates Associated Across Applications

All the widgets in a custom dashboard template that is associated with a tier or node in a different application must use the default baseline, if they use a baseline.

To set a widget’s baseline to the default baseline, or to verify that it is set, in the Select Baseline section of the Advanced panel of the widget Edit Series window, select Default Baseline.

Manage Custom Dashboard Templates

To perform miscellaneous operations on dashboard templates:

1. In the dropdown list click Manage Dashboards.
2. From the Tier or Node Dashboards window you can:
   • Delete, copy, import, export a selected dashboard template
   • Create a new dashboard template
   • Associate an existing template with tiers or nodes in the application
   • View and edit an existing dashboard template
Import and Export Custom Dashboards and Templates Using the UI

On this page:
- Export Custom Dashboards
- Export Custom Dashboard Templates
- Import Custom Dashboards
- Import Custom Dashboard Templates

Related pages:
- Configuration Import and Export API

Exporting a custom dashboard or custom dashboard template generates a JSON file that you can import into another AppDynamics Controller.

You can import or export custom dashboards and templates from the Controller UI, as described here, or programmatically with the REST API. See Configuration Import and Export API for importing and exporting programmatically.

Export Custom Dashboards

To export a custom dashboard in the Controller UI:

- From the custom dashboard list, select the dashboard that you want to export and click Export in the Dashboards menu bar.
- From the dashboard editor of the dashboard that you want to export while in edit mode, select Export Dashboard from the More Actions, lightning bolt, dropdown list.

If you have edit permissions for dashboards, you can use either method. If you do not have edit permissions, use the Export option from the custom dashboard list.

The dashboard is exported to a file named `CustomDashboard_<dashboard_name>_<unique_id>.json`. Optionally, you can rename the file before importing it.

Export Custom Dashboard Templates

After creating a custom dashboard, you can export it as a template. Templates let you quickly create new dashboards based on the dashboard.

To export a custom dashboard template, click Export button in the toolbar of the custom dashboard.

Importing and exporting are for re-using the template in a different application or a different controller. If you want to use the template in another tier or node in the same application you do not have to export and import it. You can simply associate it. See Associate a Custom Dashboard Template with a Tier or Node in Create and Manage Custom Dashboards and Templates.

Import Custom Dashboards

You can import a dashboard file to create a new dashboard based on a previously exported one.

1. From the custom dashboards list, click Import in the menu bar
2. Click Choose File and navigate to the previously exported JSON file that you want to import.
3. Click Open.
4. Click Import.

If the metrics, nodes, tiers or applications used in the exported dashboard do not exist in the new Controller environment, warning messages are displayed.

The warnings look like this:
Even when the metrics are not available in the new Controller environment, your import succeeds and preserves the widgets and layout of the dashboard. You can modify the widget properties substituting different metrics to create a similar-looking dashboard in the new environment.

Import Custom Dashboard Templates

1. In the pulldown menu of the My Dashboards tab of the tier or node dashboard from which you want to import a template click Manage Dashboards.
2. In the Tiers/Nodes Dashboards list of dashboard templates, select the dashboard template that you want to import and click Import in the menu bar.
Dashboard Recovery

Whenever a user is deleted from the user list, any custom dashboard that the user has created becomes non-functional with no permissions, and is unable to execute properly. This impacts all associated dashboard-based reports and shares. We can solve this problem using the Recover function.

Recovering a non-functional dashboard reassigns ownership to the recovering user which automatically makes the custom dashboard, its shares, and related reports functional again. The Recover function can be accessed by right-clicking the required dashboard available in the dashboard listing, and then selecting the Recover option from the context menu.

| Recover option is activated only when the dashboard owner is a deleted user. |

Use the following steps to recover a non-functioning dashboard:

1. Login to the Controller as a user with VIEW_DASHBOARDS and CREATE_DASHBOARDS permissions.
2. Access the custom dashboards listing page.
3. Select the non-functioning dashboard and right-click to access the context menu.
   
   When the owner of the dashboard is active, the Recover option remains disabled as shown below:

   ![Dashboard Context Menu](image)

   When the owner is deleted, the Recover option is enabled automatically.
4. Choose **Recover** from the context menu. The following confirmation dialog box appears in case of single dashboard recovery:

By recovering this dashboard, you will enable all existing shares and reports to be functional again. You will become the new owner of this dashboard and your view of the dashboard will be reflected in all existing shares and reports.

You can select multiple non-functional dashboards and recover all of them at once. To select multiple custom dashboards, use control-click for non-contiguous rows and shift-click for contiguous rows. Also, you must ensure that the owners of all the selected dashboards are deleted users, else the Recover function remains disabled.

5. Once the dashboard is recovered successfully the following message pops up on the screen:
To recover a dashboard, a user must have the `CREATE_DASHBOARD` permission. Whenever a user executes the Recover function without this permission, the following message is displayed:

![Dashboard recovery failed](image)

Duplicating a dashboard creates a new dashboard with the same capabilities as the non-functioning dashboard; however, duplication does not make the shares or the associated reports. The result is that the new dashboard operates using deleted user's permissions (as with Recover), but the shares and reports have to be manually recreated.

In some cases, duplicating the inactive dashboard helps customers in regaining control over the shares and reports. However, there are cases when the customers depend on the reports and shares created by the deleted user where duplication would not work. This is where the recover function is a better choice which retains all existing shares and related reports.
Reports

On this page:
- Scheduled Reports
- Other Reports

AppDynamics can extract data from dashboards and create scheduled reports. Configure automatic report generation on a regular schedule or create reports manually when needed.

Scheduled Reports

Scheduled reports are created automatically on a regular interval. AppDynamics helps create reports with the data pulled out from Dashboards and sends it to the configured list of email recipients, as scheduled. To deliver a report instantly, use Send Report Now option available for a report.

You must have account-level permissions to view and configure scheduled reports. See account-level permissions.

Report Types and Formats

Schedule periodic generation and delivery of the following report types to a list of email recipients:

<table>
<thead>
<tr>
<th>Report Type</th>
<th>Captured Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Health Report</td>
<td>Application health data from the application dashboard that includes:</td>
</tr>
<tr>
<td></td>
<td>• Business transaction and server health data</td>
</tr>
<tr>
<td></td>
<td>• Transaction scorecard</td>
</tr>
<tr>
<td></td>
<td>• Events and exceptions</td>
</tr>
<tr>
<td></td>
<td>• Load, response time, and errors graphs</td>
</tr>
<tr>
<td>Dashboard Report</td>
<td>Dashboard data in the selected custom dashboard</td>
</tr>
<tr>
<td>Controller Audit Report</td>
<td>Audit entries that include:</td>
</tr>
<tr>
<td></td>
<td>• User logins and information changes</td>
</tr>
<tr>
<td></td>
<td>• Controller configuration changes</td>
</tr>
<tr>
<td></td>
<td>• Application properties and object changes such as policies, health rules, EUM configuration</td>
</tr>
<tr>
<td></td>
<td>• Environment properties changes</td>
</tr>
<tr>
<td></td>
<td>For Controller Audit Reports, AppDynamics supports the following file formats:</td>
</tr>
<tr>
<td></td>
<td>• PDF</td>
</tr>
<tr>
<td></td>
<td>• JSON</td>
</tr>
<tr>
<td></td>
<td>• CSV</td>
</tr>
<tr>
<td>Home Screen Report</td>
<td>Overview data from the home page</td>
</tr>
<tr>
<td>All Applications Summary</td>
<td>Data summary from the applications page</td>
</tr>
<tr>
<td>User Experience: Browser Apps</td>
<td>Browser RUM data from the Browser App Dashboard Overview page</td>
</tr>
</tbody>
</table>

Manage Scheduled Reports

You can create and get schedule reports delivered on a regular interval. To create a report identical to the existing one, you can quickly duplicate the existing scheduled report and modify relevant fields, as required. Add a set of email recipients for the report delivery.

In addition to the scheduled delivery of reports, use the Send Report now option to send the generated report immediately to the email...
recipients. On the Reports page, select your report from the list and click Send Report Now from the right-click menu.

To create a scheduled report:

You need to have the account-level Configure Scheduled Reports permission to create scheduled reports.

2. On the Create Scheduled Report page, customize your report as follows:
   a. Enter Report Title and Report Subtitle.
      Tip: You can label a report CONFIDENTIAL using Report Subtitle.
   b. Optionally, select Show Title Page to include a title page at the beginning of your report file.
   c. Select your Report Type from the drop-down list. Based on this selection, fields in the Report Data tab vary.
   d. Make necessary changes in the Schedule, Report Data, and Recipients tabs and click Save.

Alternatively, you can click Create Report from the Dashboards & Reports page to view Create Scheduled Report page.

To create a controller audit scheduled report:

Create a Controller Audit report to view changes made to the user information, controller configuration, and application properties. Following are the attributes used in a controller audit report to organize audit data:

- Date/time
- Username
- Action
- Application Name
- Object type
- Object name
- API key name, if applicable
- API key ID, if applicable

These attributes can also be used as Include/Exclude filters for the Scheduled Controller Audit report.

When you are creating a scheduled report, you have to choose the Report Type on the Create Scheduled Report page. Based on this selection, fields in the Report Data tab vary.

Once you select the Report Type as Controller Audit, the Reports Data tab allows you to do the following:

- Include data from a standard or custom time ranges. You can create and manage custom time range, if required.
  Note: Custom time range options are available for all the Report Types.
- Select your report file format:
  - Specify the file format of your scheduled controller audit report such as PDF, JSON and CSV.
  - Optionally, uncheck the Show Diff box to remove the Object Changes column from your report file.
- Select an attribute from the drop-down list, enter the attribute value for filtering, and click +Add.

Use exclude and include filters on the data that appears in a controller audit report file (such as Date/Time, Username, Action, Application Name, Object Type, Object Name, API key Name, and API key ID) to manage specific audit entries in the report.
### To view, edit, duplicate, enable, disable, export, import, and delete a scheduled report:

You can manage scheduled reports from Dashboards & Reports > Reports > Scheduled Reports page. To edit, duplicate, enable, disable, export, import and delete reports, use the menu bar on the Scheduled Reports page or the right-click menu options available for each report in the list.

Ensure that you have enabled the report if you want to create and send the scheduled report on a regular interval. Disable a scheduled report when want to put it on hold for some time. You can verify the current status of a report using the Enabled column in the list of reports. For any report in the list, you can use Send Report Now option to generate and deliver it instantly.

### Other Reports

Other Report types that you can generate using Controller UI are:

1. A report on metrics data over a certain time range:
   a. Select the report in the Metric Browser with the metrics to be reported.
   b. Select the desired time range.
   c. Click Export Data and select Export as PDF Report.

2. A report on Call graph data:
   a. In the lower-left corner of the call drill-down window of a transaction snapshot, click Export to PDF.

In case if a customer wants to create a report instantly then:

1. Create a custom dashboard that displays the desired data
2. Go to scheduled reports and create a report based on the created dashboard
3. Set the report to be sent one time only or at regular intervals

### Alternative

You can use the controller audit log that replicates the Controller Audit Report. This audit log helps monitor user activities and configuration changes in the Controller. See Controller Audit Log.