AppDynamics App iQ Platform
AppDynamics Platform
Version 4.5.13
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Analytics</td>
<td></td>
</tr>
<tr>
<td>Overview of Application Analytics</td>
<td>4</td>
</tr>
<tr>
<td>Analytics Data Sources</td>
<td></td>
</tr>
<tr>
<td>Administer the Analytics Agent</td>
<td>11</td>
</tr>
<tr>
<td>Analytics Deployment Options</td>
<td>12</td>
</tr>
<tr>
<td>Installing Agent-Side Components</td>
<td>16</td>
</tr>
<tr>
<td>Upgrade Analytics Agent</td>
<td>34</td>
</tr>
<tr>
<td>Analytics and Data Security</td>
<td>35</td>
</tr>
<tr>
<td>Managing API Keys</td>
<td>37</td>
</tr>
<tr>
<td>Managing Field Visibility</td>
<td>40</td>
</tr>
<tr>
<td>Manage Business Journeys and Experience Levels</td>
<td>44</td>
</tr>
<tr>
<td>View Analytics Agent Health</td>
<td>48</td>
</tr>
<tr>
<td>Application Analytics Licenses</td>
<td>51</td>
</tr>
<tr>
<td>Remote Analytics Agent Sizing</td>
<td>54</td>
</tr>
<tr>
<td>Enable SSL for the Analytics Agent</td>
<td>58</td>
</tr>
<tr>
<td>Troubleshooting Analytics Agent Issues</td>
<td>60</td>
</tr>
<tr>
<td>Analytics Agent Logging</td>
<td>65</td>
</tr>
<tr>
<td>Configuring Application Analytics</td>
<td>66</td>
</tr>
<tr>
<td>Collect Transaction Analytics Data</td>
<td>67</td>
</tr>
<tr>
<td>Add Custom Fields to Transactions Using Java SDK</td>
<td>74</td>
</tr>
<tr>
<td>Collect Business Data From SQL Calls</td>
<td>77</td>
</tr>
<tr>
<td>Collect Log Analytics Data</td>
<td>81</td>
</tr>
<tr>
<td>Configure Log Analytics Using Source Rules</td>
<td>82</td>
</tr>
<tr>
<td>Field Extraction for Source Rules</td>
<td>90</td>
</tr>
<tr>
<td>Agent Scopes for Configuration Log Management</td>
<td>93</td>
</tr>
<tr>
<td>Collect Log Analytics Data from Syslog Messages</td>
<td>96</td>
</tr>
<tr>
<td>Configure Log Analytics Using Job Files</td>
<td>101</td>
</tr>
<tr>
<td>Sample Log Analytics Job Files</td>
<td>114</td>
</tr>
<tr>
<td>Migrate Log Analytics Job Files to Source Rules</td>
<td>115</td>
</tr>
<tr>
<td>Business Transaction and Log Correlation</td>
<td>117</td>
</tr>
<tr>
<td>Using Application Analytics Data</td>
<td>125</td>
</tr>
<tr>
<td>Search Analytics Data</td>
<td>127</td>
</tr>
<tr>
<td>Create Advanced Analytics Searches</td>
<td>131</td>
</tr>
<tr>
<td>Create Basic Analytics Searches</td>
<td>133</td>
</tr>
<tr>
<td>Investigate Using Relevant Fields</td>
<td>140</td>
</tr>
<tr>
<td>Visualize Analytics Data</td>
<td>143</td>
</tr>
<tr>
<td>Visualize JavaScript Errors</td>
<td>152</td>
</tr>
<tr>
<td>Create Analytics Metrics From Scheduled Queries</td>
<td>155</td>
</tr>
<tr>
<td>Business Journeys</td>
<td>158</td>
</tr>
<tr>
<td>Configure Business Journeys</td>
<td>163</td>
</tr>
<tr>
<td>View Business Journeys</td>
<td>168</td>
</tr>
<tr>
<td>Create an Example Business Journey</td>
<td>173</td>
</tr>
<tr>
<td>Experience Level Management</td>
<td>178</td>
</tr>
<tr>
<td>Configure Experience Level Management</td>
<td>182</td>
</tr>
<tr>
<td>View and Export XLM Compliance Data</td>
<td>185</td>
</tr>
<tr>
<td>Migrate XLM Configurations Between Environments</td>
<td>189</td>
</tr>
<tr>
<td>ADQL Reference</td>
<td>190</td>
</tr>
<tr>
<td>ADQL Typographical Conventions and Symbols</td>
<td>191</td>
</tr>
<tr>
<td>ADQL Queries</td>
<td>192</td>
</tr>
<tr>
<td>SELECT Clause</td>
<td>193</td>
</tr>
<tr>
<td>AS Clause</td>
<td>194</td>
</tr>
<tr>
<td>FROM Clause</td>
<td>195</td>
</tr>
<tr>
<td>WHERE Clause</td>
<td>196</td>
</tr>
<tr>
<td>GROUP BY</td>
<td>197</td>
</tr>
<tr>
<td>ORDER BY Clause</td>
<td>201</td>
</tr>
<tr>
<td>HAVING Clause</td>
<td>203</td>
</tr>
<tr>
<td>SINCE...UNTIL Clause</td>
<td>206</td>
</tr>
<tr>
<td>LIMIT Clause</td>
<td>207</td>
</tr>
<tr>
<td>ADQL Expressions</td>
<td>209</td>
</tr>
<tr>
<td>Analytics Functions</td>
<td>211</td>
</tr>
<tr>
<td>Analyzed Fields</td>
<td>220</td>
</tr>
<tr>
<td>Math Expressions</td>
<td>223</td>
</tr>
<tr>
<td>Comparison Operators</td>
<td>225</td>
</tr>
<tr>
<td>Logical Operators</td>
<td>228</td>
</tr>
<tr>
<td>REGEXP Operator</td>
<td>229</td>
</tr>
<tr>
<td>ADQL Data</td>
<td>231</td>
</tr>
<tr>
<td>Analytics Transaction Data</td>
<td>232</td>
</tr>
<tr>
<td>Analytics Log Data</td>
<td>235</td>
</tr>
<tr>
<td>Analytics Custom Events Data</td>
<td>236</td>
</tr>
<tr>
<td>Analytics Browser Requests Data</td>
<td>237</td>
</tr>
<tr>
<td>Analytics Browser Sessions Data</td>
<td>239</td>
</tr>
<tr>
<td>Analytics Mobile Crash Reports Data</td>
<td>241</td>
</tr>
</tbody>
</table>
Analytics Mobile Requests Data ......................................................... 243
Analytics Mobile Sessions Data ......................................................... 245
Analytics Mobile Non-Fatal Issues Data ............................................. 248
Analytics Synthetic Sessions Data ......................................................... 250
Analytics Connected Device Data ......................................................... 252
Application Analytics

AppDynamics Application Analytics powers Business iQ, a powerful part of the AppDynamics Application Performance Management (APM) Platform. Integration of all of AppDynamics APM performance engines results in a dynamic, accessible set of data about every business transaction. Business iQ can extract that information and generate baselines and dashboards, and give you perspective beyond traditional APM, enabling you to do real-time analysis of business performance that is correlated with the performance of your application software.

You can use Application Analytics with the APM, Browser RUM, Mobile RUM, and Browser Synthetic Monitoring product modules to generate:

- Transaction Analytics
- Log Analytics
- Browser Analytics
- Mobile Analytics
- Browser Synthetic Analytics

---

Installation and Administration

- Analytics Deployment Options
- Analytics and Data Security
- Upgrade Analytics Agent
- Upgrade Platform Components

Analytics Data Collection

- Collect Log Analytics Data
- Collect Transaction Analytics Data
- Business Transaction and Log Correlation
- Collect Business Data From SQL Calls

Using Application Analytics

- Business Journeys
- Experience Level Management
- Search Analytics Data
- Visualize Analytics Data

Extensibility and Reference

- Create Analytics Metrics From Scheduled Queries
- Analytics Events API
- ADQL Reference
Overview of Application Analytics

On this page:

- Analytics Home Page
- Home Page Data
- Using Home Page Data
- Restrictions and Security

Application Analytics can help you answer business-oriented questions, such as:

- How many users experienced failed checkout transactions in the last 24 hours?
- How much revenue was lost because of these failures? How is that revenue distributed across different product categories?
- What is your revenue for the day for a geographical region?
- What was the revenue impact by product category associated with two marketing campaigns we ran last week?

Analytics Home Page

The Analytics Home page consolidates data from transaction, browser, and mobile events. The home page automatically generates Transaction and End User Monitoring Summary panels through queries that aggregate data into widgets.

No configuration is required to see the Analytics Home page. However, you need appropriate licenses and access to view the widgets. See Restrictions and Security for more details.

You can access the home page at any time by clicking the Home icon on the left navigation pane in Analytics.

Home Page Data
AppDynamics Query Language (ADQL) queries generate the data for each home page widget. You cannot access or edit the ADQL queries.

Your selected time-range automatically applies a time filter to each query. Widgets refresh every two minutes.

**Transaction Summary**

The **Transaction Summary** panel aggregates your Analytics data from applications and business transactions. The **Transaction Summary** widgets are categorized by transaction threshold, from normal to error. The **Transaction Summary** panel contains six widgets:

- **Total Transactions** - count of all transactions
- **Problematic Transactions** - count of transactions that are not within normal threshold boundaries
- **Normal Transactions** - count of transactions that fall within the normal threshold boundaries
- **Top 10 Applications by Transaction Volume** - graphical view of applications
- **Top 10 Business Transactions by Transaction Volume** - graphical view of transactions
- **Transaction Score Trend** - total number of transactions for all applications

**End User Session Summary**

The **End User Session Summary** panel pulls together browser and mobile session data into widgets. The **End User Session Summary** panel contains six widgets:

- **Browser Sessions** - count of unique browser sessions
- **Mobile Sessions** - count of unique mobile sessions
- **Browser & Mobile Users** - count of browser and mobile users
- **Browser Session Bounce Rate** - rate of bounces per total sessions
  A bounce is a browser session where the page count equals one. Bounces occur when a user lands on a webpage and leaves the site without visiting additional pages.
- **Top Page Referrers by Session Volume** - the website visited by your user before accessing your site
  Referrer is an optional component of the Events Service. If you have not configured referrers, this widget does not contain data.
- **Session Trend** - total number of sessions separated by browser and mobile sessions

**Using Home Page Data**

You can select the transaction score in each chart to isolate metrics. For example, hovering over Normal in **Transaction Score Trend** removes other transaction scores, as demonstrated below.
In graph widgets, you can hover over any point to see a count of the data, as shown below.

Restrictions and Security

If your Analytics Home page does not display data on widgets, there may be an error in fetching data or no data available in the Events Service.

If panels or widgets are missing from your home page, confirm that you have sufficient licenses and have access to Analytics data.

**Licenses**

To see data on the **Transaction Summary** panel, you need:

- At least one application enabled for Analytics
- Active Transaction Analytics licenses
  
  If your Transaction Analytics license expires, Analytics stops ingesting data for the Analytics Home page. The **Transaction Summary** panel still displays your existing data, but does not ingest any new data.

To see data in the **End User Session Summary** panel you need:

- Browser Analytics and/or Mobile Analytics enabled. See Analytics Data Sources.
Active EUM licenses to Mobile and/or Browser Real User Monitoring

Some widgets may not be available depending on your End User Monitoring (EUM) licenses. For example, if you have a license to Mobile RUM only, your Analytics Home page will not contain browser-specific widgets (browser sessions, bounce rate, and referrers). Aggregated widgets, Browser & Mobile Users and Session Trend, only show the data to which you have an EUM license.

**RBAC**

Analytics enforces role-based access control (RBAC). As a result of RBAC, your Analytics Home page displays only the data to which you have access.

In the **End User Session Summary** panel, numeric widgets and the **Session Trend** chart require access to both Browser and Mobile Real User Monitoring. To see data in these widgets, your role must include access to **Browser Requests** and **Mobile Requests and Sessions** events.

See Analytics and Data Security for more information.
Application Analytics can collect data from many sources in your AppDynamics APM Platform, as listed in this topic.

### Transaction Analytics Data Sources

Transaction Analytics data is collected by the AppDynamics Java, .NET, PHP and Node.js App Agents. Collecting Transaction Analytics data requires no change to your application code. You enable analytics on the app server agents and Controller you already use.

Once you have enabled analytics on your application, you can collect and analyze several kinds of data:

- Default performance data collected by the app agents about your application's business transactions
- HTTP-based data collected by HTTP data collectors
- Custom data collected by method invocation data collectors
- Business data collected from SQL calls parameterized

To enable Transaction Analytics for your applications, see [Collect Transaction Analytics Data](#).

### Log Data Sources

You can collect Log Analytics data from many types of log files, including instrumented and non-instrumented applications as well as infrastructure. Log Analytics can be used as a standalone component. You can search and analyze log data just as you do transaction data. Log Analytics works by default with the syslog (log4j) format and can be configured for other log formats, including GZIP files (log files ending in .gz). You can capture other log formats by setting up regular-expression-based mapping. See [Collect Log Analytics Data](#) for details.

You can view correlated Log and Transaction Analytics Data. By configuring business transaction GUID injection, you can see logs that are related to specific business transaction requests. For details on configuring GUID injection, see [Business Transaction and Log Correlation](#).

### End User Monitoring Data Sources

Analytics can also use data collected by [Browser Real User Monitoring](#), [Mobile Real User Monitoring](#) and [IoT Monitoring](#). Analytics makes the data available in a more flexible search format. If you have enabled Browser RUM, Mobile RUM or IoT Monitoring and enabled Analytics, you see these event types on the Analytics search page.

For details on how analytics extends the capability of EUM, see the section "Browser Analyze versus Browser Request Analytics" in [Browser RUM Analyze](#).

### Browser Synthetic Data Sources
Sessions data captured via Browser Synthetic Analytics is available in Analytics.

**Custom Event Sources**

You can add custom events for Analytics using the AppDynamics Analytics Events API.
Administer the Analytics Agent

### Related pages:
- Analytics Deployment Options
- Installing Agent-Side Components

Application Analytics is built on the AppDynamics Application Performance Management (APM) platform, which includes the Events Service, the unstructured document store for the platform. The exact components you need to install depend upon your environment and your requirements.

- To plan your deployment, see Analytics Deployment Options.
- If you are upgrading from a previous version, see Upgrade Analytics Agent.
- If you have been using job files for Log Analytics and want to switch to using source rules (part of Configuration Log Management), see Migrate Log Analytics Job Files to Source Rules.
- To view Analytics Agents connected to your Controller, see View Analytics Agent Health.
Analytics Deployment Options

On this page:

- Analytics Agent-Side Components
- Browser and Mobile Analytics
- Deploying Analytics Agents to Multiple Nodes
- Server Side Components for On-Premise

Related pages:

- Installing Agent-Side Components
- Events Service Deployment
- Controller Deployment

When planning your deployment, consider the following questions.

Are you implementing Application Analytics as a SaaS or an on-premises solution?

- For SaaS, the Events Service is provided as part of the SaaS service. AppDynamics stores the data and hosts the server components of the system for you. You install only the Analytics Agent components of the system. This is described in the section “Agent Side Components”.
- For on-premises, you host the components yourself, storing all data on-premises. This means you need to install the Agent Side Components and the Controller and Events Service components described in Controller Deployment and Events Service Deployment. The on-premises deployment involves additional setup and administration, but it enables you to retain the analytics data within your own data center.

For each monitored machine, what data are you capturing for analytics? Do you plan to use only Log Analytics or only Browser and Mobile Analytics?

- For log data only:
  - Install either the Standalone Machine Agent or the Analytics Agent (standalone binary) on the machine.
- For both log data and transaction data:
  - Install either the Standalone Machine Agent or the Analytics Agent (standalone binary) on the machine.
  - Enable the app agent Analytics Dynamic Service.
- For transaction data only:
  - Download and install either the Standalone Machine Agent or the Analytics Agent (standalone binary) on each machine. You can also use a shared Analytics Agent instance.
  - Enable the app agent Analytics Dynamic Service.
- For Browser and Mobile Requests and Sessions data:
  - You do not need to install any agent-side Analytics components. The EUM processor passes data to Analytics automatically.

Will your Analytics Agent be installed on the same host as your application?

- If your Analytics Agent is on a host separate from the monitored application, see Remote Analytics Agent Sizing.

Analytics Agent-Side Components

For both SaaS and on-premises Application Analytics, you need to deploy (or enable) agent-side Analytics components in your application environment. These appear on the left side of the diagram.
Legend: Agent Components

1. **AppDynamics App Agent**: Application Analytics relies upon the same app server agents that AppDynamics APM uses. If you use AppDynamics APM, you likely already have these deployed to your monitored applications.

2. **Analytics Agent**: The Analytics Agent collects data from one or more app server agents and sends it to the Events Service. It also reads and transmits log data from log files from the local machine. The Analytics Agent is available as a standalone binary or as part of the Standalone Machine Agent download:
   - As part of the Standalone Machine Agent download, the Analytics Agent runs as an extension (also called a monitor)
   - A standalone binary distribution of the Analytics Agent is available for Windows environments or other environments that do not have the Standalone Machine Agent installed.

3. **Analytics Dynamic Service**: The Analytics Dynamic Service extends your app server agent functionality to collect and forward data to the Analytics Agent. It is built into the Java and .NET Agents but is not enabled by default. No additional download is needed.

### Connection Ports

For SaaS-based installations, the connection between the Analytics Agent and the Events Service in the cloud takes place over ports 80 (HTTP) and 443 (HTTPS).

For on-premises installations, the port on which the Events Service receives data from the Agent is configured during installation.

### Browser and Mobile Analytics

To see Analytics data from Browser and Mobile Real User Monitoring, you need to enable those components and enable Analytics. For information on using the on-premises EUM server, see [EUM Server Deployment](#).

### Deploying Analytics Agents to Multiple Nodes

A real-world scenario is unlikely to consist of a single monitored node, as shown in the diagram above. It usually consists of many applications deployed over many hosts.
While the APM app server agents continue to send data to the Controller in the normal way, the Analytics Dynamic Service sends its
data to the Analytics Agent. This agent runs in a separate JVM process in the local environment or network, either as part of the
Standalone Machine Agent or on its own.

There must be at least one Analytics Agent in the monitored environment, although multiple app server agents collecting only
transaction data can share a single Analytics Agent, as shown in the figure. However, each machine where you want to gather log data
must have its own Analytics Agent instance.

In this example, AppServerHost1-3 are collecting only transaction data, so only app server agents are required. Each one connects to
the Analytics Agent on AppServerHost4, where log information is also being collected. The Analytics Agents could also reside on the
same machines as each app server agents. On AppServerHost5, only log data is being collected, so an Analytics Agent is also
necessary there.

Server Side Components for On-Premise

To set up AppDynamics Analytics on-premise, you also need to install the server parts of the system, the AppDynamics Controller and the Events Service.

The following figure depicts the basic components of an On-premises deployment:

Legend: Server Components

4 AppDynamics Controller: The heart of an AppDynamics deployment, the Controller processes and presents the information gathered by the agents.

5 Events Service: The unstructured document store, it gathers and stores data from the Analytics, Database, and Network Visibility Agents and, if you have End User Monitoring, from the EUM Server. It allows the Controller UI to run queries on that data.

6 If you are using SaaS EUM, you must use the SaaS Events Service. If you are using the on-prem EUM Server, you must use an on-premises instance of the Events Service.
Installing Agent-Side Components

In all AppDynamics Application Analytics deployment scenarios, you must enable the Analytics Agent-side components. The Analytics Agents are distributed as operating system-based or standalone bundles which you can download from the AppDynamics Releases page. This section describes different scenarios of setting up each component of Application Analytics to collect business transaction data.

Prepare to Enable the Agents

Review the deployment options and the agent-side component architecture in Analytics Deployment Options to confirm which components you need to install.

Analytics Agent

The Analytics Agent collects data from AppDynamics App Agent (app server agent) and log files, and sends it to the Events Service. The Analytics Agent is not enabled by default. To use the Analytics Agent, you need to download the desired bundle and enable it using a method of your choice as described in this topic. To use Application Analytics, you must:

1. Obtain a separate Application Analytics license.  
   See Application Analytics Licenses for more information.
2. Enable the Analytics Agent.
3. Point to the Event Service.  
   See Revise the Analytics Agent Properties File for information on agent-specific configuration.
4. Enable Analytics on the Controller.

AppDynamics App Agent

In all environments, to collect transaction data from an application, you must have a supported app server agent deployed. If you already use AppDynamics APM, agents might already be installed in your environment. To access the Analytics functionality, the app server agent must be version 4.0 or later. For detailed information for installing the app server agents, see the following topics:

- Java Agent
- .NET Agent
- Node.js Agent

Install Analytics Agent on Windows

As described in Analytics Deployment Options, the Analytics Agent can run on the same host or on a different host from the app server agent, depending on your use case. The AppDynamics Releases page provides the following distribution archives:

- Standalone Analytics Agent (no JRE)
- Analytics Agent with JRE 1.8 for both 32-bit and 64-bit Windows machines.

Enable Analytics Agent for a Local App Server Agent

This section assumes that you are installing the Analytics Agent on the same host as your app server agent. In this deployment, the Analytics Agent reads and transmits log data from log files from the local machine. The app server agent transmits data from the monitored application to the Analytics Agent.
You install the Analytics Agent using the .exe file, analytics-agent.exe, found in the `<analytics-agent-home>/bin` directory. The Analytics Agent is written in Java, so you run it in a JVM. On Windows machines, the Analytics Agent runs as a Windows service.

**Installing the Analytics Agent**

1. Unzip the Analytics Agent distribution archive to the installation directory on each target host. When you unzip this archive, you get three directories:
   - `bin`: contains the binary file for Windows (.exe)
   - `lib`: contains all the jar files that need to be in the class path
   - `conf`: contains all the configuration files, such as the properties and vmoptions files
2. Configure the Analytics Agent by modifying the `analytics-agent.properties` file:

   `<analytics-agent-home>/conf/analytics-agent.properties`

   For more information on configuration instructions, see Revise the Analytics Agent Properties File.
3. If you are collecting log information on this host, you need to:
   - Configure your log sources. See Collect Log Analytics Data for details.
   - Update additional properties. See Configure Properties for Consolidated Log Management.
4. Save and close the file.
5. Run the following command to install the Analytics Agent as a service.

   `bin\analytics-agent.exe service-install`

   Now your analytics-agent can be managed like any other Windows service.
6. Start the service using the method of your choice as described in Starting and Stopping the Analytics Agent.

**Revise the Analytics Agent Properties File**

The Analytics Agent properties file determine how your Analytics Agent communicate with other components, what type of data it gathers from the monitored application, and how the components of the Application Analytics deployment and the Controller authenticate to each other. You modify the default configuration and set desired values for the agent properties by using the Analytics Agent properties file. Note the differences in the location of this file depending on your exact deployment scenario.
1. Open the `<analytics-agent-home>/conf/analytics-agent.properties` file using a text editor.
2. Change the default URL and, if necessary, the port number for the connection to the Events Service by modifying the `http.event.endpoint` value. For example:

   ```
   http.event.endpoint=http://<events_service_host>:events_service_port>
   ```

   For SaaS-based installations, the host and port combination is one of the following:
   - https://analytics.api.appdynamics.com:443 (North America)
   - https://fra-ana-api.saas.appdynamics.com:443 (Europe)
   - https://syd-ana-api.saas.appdynamics.com/:443 (APAC)

   If your firewall rules use IP addresses, review the "Firewall Considerations" section in the Troubleshooting Analytics Agent Issues topic.

   For on-premises installations use whatever host and port you have configured. In clustered environments, this is often a load balancer.

3. Change the default URL and the port number to point to the Controller by modifying the `ad.controller.url` property. For example:

   ```
   ad.controller.url=<controller_host_name>:<http-listener-port>
   ```

   This is the URL and the port number you use to access the Appdynamics browser-based user interface.

4. Note down the Global Account Name and Access Key values found on the View License UI of the Controller.

5. Configure the account and account key for which the agent should publish the business transaction data. Use the Global Account Name and Access Key values you have collected earlier. For example:
5. # The global_account_name in the Controller for this analytics data, similar to the following:
   http.event.accountName=<customer1_74678b04-8a71-40ef-acaf-xxxxx
   xxxxxxxx>
   # Replace this value with the access key of the account name configured above.
   http.event.accessKey=<3d58aba2-xxx-xxx>

The `http.event.accountName` property specifies the **Global Account Name** of the account. The `http.event.accessKey` property specifies the **Access Key**, which provides an authentication mechanism between the Controller and the components of the Application Analytics deployment. The Controller installation process generates the **Access Key** value. For SAAS-based installations, set the `http.event.name` property to **Name** of the account. The default value is `customer1`. If the property is not configured correctly, the Analytics Agent will not be able to authenticate. The resulting errors are stored in the `analytics-agent.log` file.

6. Change these properties, which are needed to run the Analytics Agent as a Windows service:
   - `ad.dw.log.path=<analytics-agent-home>\logs`
   - `conf.dir=<analytics-agent-home>\conf`
   - `ad.jvm.options.name=analytics-agent.vmoptions`
   - `ad.jvm.heap.min=512m`
   - `ad.jvm.heap.max=1g`

   The Analytics agent on Windows requires double backslash for paths, for example:

   ```
   conf.dir=C:\AppD\analytics-agent\conf
   ad.dw.log.path=C:\AppD\analytics-agent\logs
   ```

7. If you are collecting log information on this host, you need to:
   - Configure your log sources. See **Collect Log Analytics Data** for details.
   - Update additional properties. See **Configure Properties for Consolidated Log Management**.

Starting and Stopping the Analytics Agent

You can use the native Windows services menu to start and stop the service or you can do it directly from the command line using the following commands.

1. Start the agent service:

   ```
   bin\analytics-agent.exe service-start
   ```

2. Stop the agent service:
**Uninstalling Analytics Agent**

At the command line, run the `.exe` file with the uninstall command as follows:

```
bin\analytics-agent.exe service-uninstall
```

**Enable the App Server Agent for a Remote Analytics Agent**

The default settings of the app server agent typically assume that the Analytics Agent is on the same host and uses the default port. If your Analytics Agent is on a host separate from the app server agent, as shown in the diagram below, or you have changed the default port, you need to specify the new host and port values for the app server agent.

On Windows, you specify the location of the remote Analytics Agent with an environment variable named `appdynamics.analytics.agent.url` as follows:

1. On your Windows machine, go to **System Properties > Advanced system settings > Environment Variables.**
2. Under System variables, click **New**.
   Although you can also use a User environment variable, the user under which the environment variable is set must have the
   same permissions as the user under which all the instrumented apps are running. To avoid issues with this, we recommend
   using a System environment variable approach.

   Replace `<analytics-agent-ip>` with the hostname of the Analytics Agent for your environment.
4. Restart the applicable application or process where you want the new environment variables to take effect. For w3wp in Windows, restart IIS by running `iisreset` after the changes are made to the environment variables. While it is not required that you restart your machine, the parent process that is invoking the monitored process must be restarted.

**Install Analytics Agent on Linux**

As described in Analytics Deployment Options, the Analytics Agent can run on the same host or on a different host from the app server agent, depending on your use case. The AppDynamics Releases page provides the following distribution archives.

- Standalone Analytics Agent (no JRE)
- Machine Agent Bundle - 64-bit osx (zip)
- Machine Agent Bundle - for both 32-bit and 64-bit Linux machines (rpm)
- Machine Agent Bundle - for both 32-bit and 64-bit Linux machines (zip)
- Machine Agent Bundle - for 64-bit solaris (zip)
- Machine Agent Bundle - for 64-bit solaris-sparcv9 (zip)
- Machine Agent Bundle - for 64-bit solaris-x64 (zip)

**Enable Analytics Agent as an Extension to Standalone Machine Agent**

If the Machine Agent and app server agent are running on the same machine, installing a Standalone Analytics Agent is not necessary. In environments with the Standalone Machine Agent already running, you enable and run the Analytics Agent as an extension. This section describes how to configure the Analytics Agent as an extension to the Standalone Machine Agent and run as a machine agent monitor.
1. Confirm that JRE 8 or higher is installed on the host running the Standalone Machine Agent. Analytics Agent cannot be enabled if the required version of JRE is not available on the host.

2. On the host running the Standalone Machine Agent, open `<machine-agent-home>/monitors/analytics-agent/monitor.xml` using a text editor.

3. Set the `enabled` tag to true as follows and save the file when you are finished.

   ```xml
   <monitor>
     <name>AppDynamics Analytics Agent</name>
     <type>managed</type>
     <!-- Enabling this requires JRE 8 or higher -->
     <enabled>true</enabled>
   ...
   ```

4. Open the following file to configure connectivity from the Analytics Agent to the Events Service.

   ```bash
   <machine-agent-home>/monitors/analytics-agent/conf/analytics-agent.properties
   ```

5. In the `analytics-agent.properties` file, do the following:
   - Change the default URL and, if necessary, the port number for connecting to the Events Service by modifying the `http.event.endpoint` value. For example:
5. Change the default URL and the port number to point to the Controller by modifying the `ad.controller.url` property. For example:

```
http.event.endpoint=http://<events_service_host:events_service_port>
```

For SaaS-based installations, the host and port are:

- https://analytics.api.appdynamics.com:443 (North America)
- https://fra-ana-api.saas.appdynamics.com:443 (Europe)
- https://syd-ana-api.saas.appdynamics.com:443 (APAC)

If your firewall rules use IP addresses, review firewall considerations in Troubleshooting Analytics Agent Issues.

For on-premises installations use whatever host and port you have configured. In clustered environments, this is often a load balancer.

6. Note down the Global Account Name and Access Key values found on the View License UI of the Controller.

![License](image)

7. Configure the account and account key for which the agent should publish the business transaction data. Use the Global Account Name and Access Key values you have collected earlier. For example:
# The global_account_name in the Controller for this analytics data, similar to the following:

```
http.event.accountName=<customer1_74678b04-8a71-40ef-acaf-xxxxx
xxxxxxx>
```

# Replace this value with the access key of the account name configured above.
```
http.event.accessKey=<3d58aba2-xxx-xxx>
```

The `http.event.accountName` property specifies the **Global Account Name** of the account. The `http.event.accessKey` property specifies the **Access Key**, which provides an authentication mechanism between the Controller and the components of the Application Analytics deployment. The Controller installation process generates the **Access Key** value. For SAAS-based installations, set the http.event.name property to **Name** of the account. The default value is customer1. If the property is not configured correctly, the Analytics Agent will not be able to authenticate. The resulting errors are stored in the Analytics-agent.log file.

8. If you are collecting log information on this host, you need to:
   - Configure your log sources. See [Collect Log Analytics Data](#) for details.
   - Update additional properties. See [Configure Properties for Consolidated Log Management](#).

9. Save and close the file.

10. If the machine-agent is already running at this point, restart it to pick up the changes in the configuration.

```
To connect to the Events Service through a proxy server, see Connect the Agent to the Events Service through a Proxy.
```

## Enable Standalone Analytics Agent

In this deployment, the Analytics Agent and app server agent are running on the same machine. Analytics Agent reads and transmits log data from log files on the host machine. The app server agent transmits data from the monitored application to the Analytics Agent.

For environments that don't have a Machine Agent installed, install the Analytics Agent as a separate binary, `analytics-agent.sh`:

1. Unzip the Analytics Agent distribution archive to the installation directory on each target host.
   When you unzip this archive, you get three directories:
   - bin: contains the binary file for Linux, Solaris, and OSX
   - lib: contains all the jar files that need to be in the classpath
   - conf: contains all the configuration files, such as the properties and vmoptions files
2. Open the following file to configure connectivity from the Analytics Agent to the Events Service.
3. In the `analytics-agent.properties` file, do the following:
   - Change the default URL and, if necessary, the port number for connecting to the Events Service by modifying the `http.event.endpoint` value. For example:

   ```
   http.event.endpoint=http://<events_service_host:events_service_port>
   ```

   For SaaS-based installations, the host and port are:
   - `https://analytics.api.appdynamics.com:443` (North America)
   - `https://fra-ana-api.saas.appdynamics.com:443` (Europe)
   - `https://syd-ana-api.saas.appdynamics.com:443` (APAC)

   If your firewall rules use IP addresses, review firewall considerations in Troubleshooting Analytics Agent Issues.

   For on-premises installations use whatever host and port you have configured. In clustered environments, this is often a load balancer.
   - Change the default URL and the port number to point to the Controller by modifying the `ad.controller.url` property. For example:

   ```
   ad.controller.url=http://<application_server_host_name>:<http-listener-port>
   ```

   This is the URL and the port number you use to access the Appdynamics browser-based user interface.

4. Note down the Global Account Name and Access Key values found on the View License UI of the Controller.

5. Configure the account and account key for which the agent should publish the business transaction data. Use the Global Account Name and Access Key values you have collected earlier. For example:
# The global_account_name in the Controller for this analytics data, similar to the following:
http.event.accountName=<customer1_74678b04-8a71-40ef-acaf-xxxxxxxx>
# Replace this value with the access key of the account name configured above.
http.event.accessKey=<3d58aba2-xxx-xxx>

The http.event.accountName property specifies the Global Account Name of the account. The http.event.accessKey property specifies the Access Key, which provides an authentication mechanism between the Controller and the components of the Application Analytics deployment. The Controller installation process generates the Access Key value. For SAAS-based installations, set the http.event.name property to Name of the account. The default value is customer1. If the property is not configured correctly, the Analytics Agent will not be able to authenticate. The resulting errors are stored in the analytics-agent.log file.

6. If you are collecting log information on this host, you need to:
   - Configure your log sources. See Collect Log Analytics Data for details.
   - Update additional properties. See Configure Properties for Consolidated Log Management.

7. Save and close the file.

8. Install the Analytics Agent by running the following command.

   ```
   bin\analytics-agent.sh start
   ```

Configure the App Server Agent for a Remote Analytics Agent

The default settings of the app server agent typically assume that the Analytics Agent is on the same host and uses the default port. If your Analytics Agent is on a host separate from the app server agent, as shown in the diagram below, or you have changed the default port, you need to specify the new host and port values for the app server agent.
Specify the location of the remote Analytics Agent with a -D parameter. The argument is passed to the monitored application as follows:

```
```

Replace `<analytics-agent-ip>` with the hostname of the Analytics Agent for your environment.

**Run the Analytics Agent From a Read-Only Filesystem**

In this deployment, the Analytics Agent is installed and run from a read-only filesystem. To do so, you need to direct the Analytics Agent to write its log files to a writeable partition.

Create a directory for the Analytics Agent in a writeable file system and create symlinks to the contents of the analytics agent in the
read-only file system, with the end result that you have a writeable top-level directory for the Analytics Agent and a writeable logs directory under that top-level directory.

To run analytics-agent from a separate writable directory using artifacts from a read-only directory:

1. Provide necessary permissions to the read-only Analytics Agent directory.

   ```
   # analytics-agent-readonly refers to the read-only analytics agent.
   chmod -R 555 /tmp/analytics-agent-readonly/
   ```

2. Create a directory for the Analytics Agent in a writable file system.

   ```
   mkdir /tmp/analytics-agent-writable
   chmod 755 analytics-agent-writable/
   cd /tmp/analytics-agent-writable
   ```

3. Create symlinks to the contents of the Analytics Agent in the read-only file system.

   ```
   ln -s /tmp/analytics-agent-readonly/bin/ bin
   ln -s /tmp/analytics-agent-readonly/lib/ lib
   ln -s /tmp/analytics-agent-readonly/monitor.xml monitor.xml
   ```

4. Create a writeable conf directory under that top-level directory.

   ```
   cd /tmp/analytics-agent-writable
   mkdir conf
   mkdir conf/
   chmod 755 conf/
   cd conf
   ```

5. Create symlinks to the contents of the conf directory.
5.

6. Create logs directory under analytics-agent-writable directory
7. Run the Analytics Agent from the writable directory.

```
ln -s /tmp/analytics-agent-readonly/conf/analytics-agent.* .
ln -s /tmp/analytics-agent-readonly/conf/job/ job
ln -s /tmp/analytics-agent-readonly/conf/grok/ grok

cd /tmp/analytics-agent-writable

cd /tmp/analytics-agent-writable
nohup bin/analytics-agent.sh start &
```

The resulting directories look similar to:
Tune Analytics Agent

This section describes the post-installation configuration of the Analytics Agent. You learn to configure Analytics Agent to connect to an Event Service using a proxy server, to collect log files using Consolidated Log Management (CLM), to set resource usage limit, and to change Java Virtual Machine (JVM) options.

Connect to the Events Service through a Proxy

If the Analytics Agent needs to connect to the Events Service through a proxy server, do the following:

1. Open `<analytics-agent-home>/conf/analytics-agent.properties` with a text editor.
2. Add this information:
# optional proxy properties
http.event.proxyHost=<your proxy host>
http.event.proxyPort=<your proxy port>
http.event.proxyUsername=<your proxy username, if authentication is required>
http.event.proxyPassword=<your proxy password, if authentication is required>

3. Save and close the file.

Configure Analytics Agent for Consolidated Log Management

If you want your Analytics Agent to work with CLM you need to configure a few properties in the Analytics Agent properties file. You provide the Controller URL, customer name, and name of your Analytics Agent in the analytics-agent.properties file. The customer name is found on the View License UI of the Controller. If you are using CLM to configure your log files with source rules, you need to provide these properties with the correct values.

The default values are the following:

# Format should be http://<host>[:<port>]
ad.controller.url=http://localhost:8090

# The customer name field from the appdynamics license page.
http.event.name=customer1

# This is the friendly agent name that will show up in the controller when the agent registers and syncs configuration.
ad.agent.name=analytics-agent1

Changing Java Virtual Machine Options
If you need to change any JVM start-up options, modify `<analytics-agent-home>/conf/analytics-agent.vmoptions` by using a text editor. Be aware that the `vmoptions` file name is read from the `ad.jvm.options.name=analytics-agent.vmoptions` properties file. So, if you change the `vmoptions` file name you need to change the `ad.jvm.options.name` property as well.

(Windows only) If the Analytics Agent Windows service is installed and you need to change the properties file or the `vmoptions` file, you need to uninstall the service and reinstall it for the changes to take effect.

### Configure Resource Usage Limit

To limit resource usage, you can enable the default limit on the number of job for a single analytics agent. This limit can be overridden but is not recommended without a thorough understanding of the potential impact on resource usages, such as CPU usage, disk, and network I/O. The property is `ad.max.enabled.jobs` and is found in the `<analytics-agent-home>/conf/analytics-agent.properties` file. By default `ad.max.enabled.jobs` is set to 20.

### Verify Analytics Agent Status

To verify that the Analytics Agent has started, you can look for the following entry in the App Agent log file: “Started [Analytics] collector”

If you need to connect to the Events Service through a proxy server, see Connect the Agent to the Events Service through a Proxy below.

### Enable Analytics on the Controller

Once you have set up the Analytics Agent, you need to enable Analytics on the Controller.

1. In the Controller UI, select the Analytics tab.
2. Select Configuration.
3. From the Transaction Analytics tab, select the application you wish to monitor from the drop-down menu.
4. Check Enable Analytics Data Collection for `<application_name>`.

### Troubleshooting Tips

- Make sure that the properties in `analytics-agent.properties` are properly set. Also, see Enabled Job Files Limit.
- JRE version is >= 1.7 and JAVA_HOME variable is set in the environment.
- All the properties in `analytics-agent/conf/analytics-agent.vmoptions` are compatible with the JRE.
Upgrade Analytics Agent

On this page:
- File Locations
- Upgrade Procedure for Analytics Agent (Standalone Binary)

Related pages:
- Migrate Log Analytics Job Files to Source Rules
- Upgrade the Standalone Machine Agent

This topic describes updating the Analytics Agent standalone binary and includes some steps that are relevant only if you are collecting Log Analytics data.

If you are running the Analytics Agent through the Standalone Machine Agent, see Upgrade the Standalone Machine Agent for instructions. Note: you should maintain a copy of the controller.xml file in addition to the analytics-agent.properties, job folder, and watermark folder.

Preserving the watermark file and the job files is only necessary when you are collecting log data for analytics. The watermark file preserves information about the number of bytes read from the log sources by the Analytics Agent and is relevant only when Log Analytics is enabled. Job files contain the configuration for pre-4.3 log sources.

File Locations

After unzipping the Analytics Agent standalone binary, the file locations are:

- property file: `<analytics-agent-home>/conf/analytics-agent.properties`
- job files: `<analytics-agent-home>/conf/job`
- watermark file: `<analytics-agent-home>/conf/watermark`

Running Analytics Agent via the Standalone Machine Agent, the file locations are:

- properties file: `<machine-agent-home>/monitors/analytics-agent/conf/analytics-agent.properties`
- job file: `<machine-agent-home>/monitors/analytics-agent/conf/job`
- watermark file: `<machine-agent-home>/monitors/analytics-agent/conf/watermark`

Upgrade Procedure for Analytics Agent (Standalone Binary)

Use these steps to upgrade your Analytics Agent (standalone binary) to 4.5.

1. Stop the Analytics Agent.
2. Download the 4.5 Analytics Agent binary from the download site.
3. Extract the new zip. Be sure to extract into a new empty directory or confirm that the old directories are empty before extracting the new version. Otherwise, you may encounter errors starting the Analytics Agent.
4. (Windows only) If the Analytics Agent is installed as a Windows Service, you need to manually uninstall the old service before installing the 4.5 version.
5. Save a copy of your old analytics-agent.properties so you can copy values to the new analytics-agent.properties file.
6. Modify the new `analytics-agent.properties` file with in accordance with the steps here: Installing Agent-Side Components, referring to your previous properties as needed.
7. If you have been collecting log analytics data:
   a. Copy the old watermark file to the corresponding location in the new `<analytics-agent-home>` directory structure.
   b. Copy your pre-existing job files to the corresponding location in the new `<analytics-agent-home>` directory structure.
8. Start the new Analytics Agent.
   You can observe that the agent starts the enabled job file, and starts tailing the logs from the last read position in the watermark file. Usually, a log message is displayed stating the start of the tailing from last read location. It matches with the watermark file. If at time printing log message gets out of sync, it can be manually validated through the UI that all logs have been tailed, and no duplicate logs have been tailed.

To move from job files to the new Consolidated Log Management source rules, see Migrate Log Analytics Job Files to Source Rules.
Analytics and Data Security

<table>
<thead>
<tr>
<th>On this page:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Roles and Permissions</td>
<td></td>
</tr>
<tr>
<td>Default Application</td>
<td></td>
</tr>
<tr>
<td>Analytics Administration UI Tabs</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Related pages:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Managing Field Visibility</td>
<td></td>
</tr>
<tr>
<td>Managing API Keys</td>
<td></td>
</tr>
<tr>
<td>Roles and Permissions</td>
<td></td>
</tr>
</tbody>
</table>

Application Analytics provides role-based access control (RBAC) to enable you to control and protect your data. The Analytics permissions enable you to provide granular, controlled access to features and analytics data in your environment. This topic outlines the available permissions.

Roles and Permissions

A predefined role called Analytics Administrator is provided with preset permissions for all Analytics-specific permissions.

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 roles and assign users to roles for Analytics, users need both the Analytics Administrator and the Account Owner role.

Also, see Business iQ Analytics Permissions.

Default Application

To ensure that your users have access to the full Analytics functionality they need, be sure to give them the view permission on the Default Application. This permission is given to all predefined admin roles and to the read-only role, Applications & Dashboards Viewer. If you create new roles for analytics, you need to grant this permission in addition to specific analytics permissions.

When you create a new role, this permission is not automatically given to that role. You can either turn on the view permission by checking the View option on Default Application or you can add the read-only role to new users.

From the Administration page Applications tab, grant view permissions to the Default Application as shown in the following screenshot:
Analytics Administration UI Tabs

Access the Analytics tab in the Controller Administration UI when creating new roles to see the available permissions.

General Permissions

This section contains permissions that control access to analytics features:

- Manage Fields
- Manage APIs
- Manage Metrics
- Configure Log Analytics with Source Rules
- Manage Business Journeys
- Manage Experience Levels

Search Permissions

All the saved searches created in Analytics appear in this section. The admin role can assign View, Edit and Delete permissions on a search by search level. The admin can create and save specific searches needed by various roles in your organization. In addition to enabling the permissions, View, Edit, and Delete, for a specific saved search, the admin must also enable access to the related application or log source type for the search. Otherwise, the data access level won't be granted to the role.

To create a new saved search, the user must have the **Can Create a Search** permission.

- **View** permission: Assigning only the View permission for a saved search means users can not edit or delete the search.
- **Edit** permission: Assigning Edit permission for a saved search means users can modify the filter criteria and save back to the same search.
- **Delete** permission: Assigning Delete permission for a saved search means users can delete the search.

Event Type Permissions

**Transactions Permissions:** This section enables the admin user to assign permissions for viewing application transaction data. Permissions can be granted to view all transaction data for the account or on an application by application basis. When creating new roles, remember that granting permissions to view transaction analytics data does not automatically grant permissions to see all application data associated with a specific transaction analytics record. You need to grant at least read-only permissions to the application to enable the user to see associated transaction snapshot data such as flow maps.

**Log Permissions:** This section enables the admin user to assign permissions for viewing log data. Permissions can be granted to view log data for all source types configured for the account or on a source by source basis.

**Browser Requests Permissions:** This section enables the admin user to assign permissions for viewing browser request data. Permissions can be granted to view data for all applications for the account or for specific applications.

**Mobile Requests Permissions:** This section enables the admin user to assign permissions for viewing mobile requests and crash report data. Permissions can be granted to view data for all applications for the account or for specific applications.

**Custom Analytics Events:** This section enables the admin user to assign permissions for querying custom analytics events data. Permissions can be granted to view data for all custom analytics events or for specific applications.

**Synthetic Permissions:** This section enables the admin user to assign permissions for querying Synthetic events data. Permissions can be granted to view data for all applications for the account or for specific applications.
Managing API Keys

The Analytics Events REST API provides three independent endpoints for the following responsibilities:

- Schema Management
- Publish Events
- Query Events

API Keys provide a secure authentication mechanism for a caller to prove identity when using these public REST APIs. Because call sites can vary across your infrastructure, departments, and geographic regions, these credentials could be widely distributed and hard to control. Publishing or querying events from different call sites, therefore, requires fine-grained control over the keys and the operations allowed for a particular key. You may also need to revoke these keys on demand if they are found to be compromised.

For these reasons, managing access for the Analytics Events API uses API keys. Your organization can create multiple API keys, manage and distribute them based on your own internal policies. For instance, each department or geographic region may be assigned their own API key for distribution and control management. If an API key is compromised, it can be deleted and a new key created without other undesirable side effects.

API keys are only valid for calling the public Analytics Events REST API and can't be used to access the AppDynamics Controller or data in any other way. For details on using the API keys, see the Analytics Events API documentation.

Permissions

API keys are used only for Analytics Events REST APIs. When you create an API key, you can specify read and write permissions for each API key.

Users with the Manage API permission can do the following actions:

- Add: Creates an API key
- Disable: Disables the calls using that key temporarily, but does not remove the key
- Enable: Re-enables a disabled key
- Delete: Removes the key permanently

It may take up to 15 minutes for a key which has been deleted or disabled to be detected and all operations using that key to be rejected.

When you create an API key, you see permissions for the various event types (logs, transactions, browser, and mobile) in our platform. By using these permissions you can limit or grant access to specific event types for each specific API key.

Examples

You can generate an API key for your partners enabling them to only publish custom events. Just check "Publish all Custom Events" checkbox when you create their API key.

You can generate an API key to enable various users to only query Apache log data. Just check the "apache" source type under "Log Permissions."

Once an API Key is created, you cannot change the permissions. You can grant permissions for API Keys as follows:

- Custom Analytics Events:
  - Manage Schema: Enables creation of schemas using the Analytics Events Schema Management APIs.
• Query Custom Events: Enables you to query all Analytics event types (with the appropriate permissions).
• Publish Custom Events: Enables you to publish Analytics custom events using the Analytics Events Publish APIs.

Transactions: Query all transactions or specific applications.
Logs: Query all logs or specific source types.
Browser Requests: Query all browser requests or specific applications.
Mobile Requests: Query all mobile requests or specific applications.
Synthetic Requests Permissions: Query all synthetic requests data or specific applications.
Connected Devices Permissions: Query all connected device requests data or specific applications.

If you have deployed EUM such that you are using an on-premises Events Service for transaction and log analytics data, and the SaaS Events Service for your EUM data, you can not query the browser or mobile request data using the Analytics API.

Create API Keys

1. From the Analytics > Configuration window, select the API Keys tab. Here you can see a list of existing keys and have access to actions for managing the keys.
2. Click +Add to see the configuration window.

3. Add a name and description and expand each permission section to select the permissions for this key.

Do NOT click Create until you have selected all the necessary permissions for your use case, including any necessary analytics data permissions because you can not change the permissions once the key is created. Once the key is created, you can only edit the description and whether the key is enabled or disabled.

4. Click Create.
You see a window containing the new key similar to the following:
5. Copy and save the key. Check the check box indicating you have copied the key and click done. You cannot retrieve the key once you dismiss this dialog.
Managing Field Visibility

This topic describes how to manage the visibility of fields in your analytics data. This capability enables you to hide fields that you don’t want other people in your organization to see.

You can set the visibility of the fields in most of the analytics events types with the exception of custom events. Hiding fields in custom events is not supported.

Permissions
The Manage Fields permission is required to manage field visibility. If a role has the Manage Fields permission, it means all the users with that role can manage all the fields they have access to. They can unhide the fields or see which fields have been hidden. For instance, if you have Manage Fields permission and view permission for the business transactions in the “Ecommerce App” application, then you can only see and manage fields for that application.

Manage Fields

1. From the Analytics Search window find and click the gear icon (⚙️) that appears near each Fields grouping to open the Manage Fields window:
2. Use the Manage Fields window to show or hide fields as needed. The list of fields varies depending on the event type selected for the search.
Delete Extracted Fields

In versions 4.3 and higher, you may have two types of extracted fields, fields created in 4.2 where the Controller version has been upgraded to a higher version and fields extracted from logs with log source rules.

For fields created in 4.2, you can delete the fields using these steps:

1. Hover (with the cursor) over the field and click on the view icon that appears.
2. Click on **Delete** in the popup.
For fields being extracted from logs using log source rules, you can delete the fields by editing the source rules. See Field Extraction for Source Rules.
Manage Business Journeys and Experience Levels

This topic describes how to manage the visibility of business journeys and experience level management (XLM) with roll-based access control (RBAC). Business journey and XLM RBAC is required to restrict access to sensitive information among your organization. If a user attempts to access restricted features, a message appears warning the user that access is denied.

Only the analytics admin can grant permissions to business journeys and XLM, as well as with their respective searches, event types, and sources. The admin authorizes access on an individual basis by assigning each user to the applicable role.

Creating Business Journey and XLM Roles

In the Controller UI, navigate to Settings > Administration > Roles.

The analytics admin can view and edit existing roles in the Analytics tab, which contains three sections: General, Searches, and Events. Select +Create to make a new role.

General Permissions

In the General tab, the analytics admin authorizes access to business journeys and/or XLM. This is required for the user to perform any actions on either business journeys or XLM. Check Manage Business Journeys and/or Manage Experience Levels, as shown below:

With general permission, the user has feature level access to business journeys and/or XLM. However, the user cannot view the definition of existing events or saved searches, or create new events, with only general permission. See the below sections to enable...
A user can access business journey data through searches without general permission to business journeys, as long as the user's role permits access to the underlying event type(s) used in business journey definition.

In this way, you can restrict access to define business journeys while also allowing a user to query the data. Ensure the user has all necessary access to event type(s) in the Events tab and leave Manage Business Journeys unchecked.

Search Permissions

There are two search types in Analytics, drag and drop and query language. The analytics admin grants access to both types in the Searches tab.

Choose "Can Create a Search" to access both search types. Choose Add to access specific saved searches.

Create search permission is required in order for a user to perform any of the following operations:

- Save a search
- Create a metric
- Create a visual widget

Search access is dependent on event type access. A user has access to create and save searches only for the event type(s) and source(s) assigned in the user's role. The next section provides instructions to assign event type and source access.

Event Type and Source Permissions

Granting permission to business journeys and/or XLM does not provide a user blanket access to all reports. Users have access only to the exact event type(s) and source(s) granted by the analytics admin. For example, transactions are an event type, and their source is applications. Therefore, if a user requires access to particular applications, the analytics admin must select transactions as well as each necessary application for the user.

The table below lists the available event types and their corresponding sources:

<table>
<thead>
<tr>
<th>Event Type</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transaction</td>
<td>Application</td>
</tr>
<tr>
<td>Log</td>
<td>Source Type</td>
</tr>
<tr>
<td>Browser Records</td>
<td>App Key</td>
</tr>
<tr>
<td>Mobile Records and Sessions</td>
<td>App Key</td>
</tr>
<tr>
<td>Synthetic Session</td>
<td>App Key</td>
</tr>
<tr>
<td>Connected Devices</td>
<td>App Key</td>
</tr>
<tr>
<td>Custom Analytics Events</td>
<td>Particular user-defined event type</td>
</tr>
</tbody>
</table>

You can specify exact access in the Events tab. Select the event type, then specify access for the relevant source. Check "Can View Data from all [source]" to grant blanket access to each source. To select individual sources, click Add.
With the appropriate event type and source permissions, the user can access existing reports as well as create new reports. For example, a user with permissions to Transactions and all applications can now create a new configuration for any application. However, if this user attempts to create a configuration for Log events, the configuration does not save and an error message appears in the UI:

For XLM configurations, the **Filter By** field indicates the source(s) of the given **Event Type**. Users with limited permission to sources need to add their permitted source(s) as filter criteria. If you do not specify filter criteria, you must have permission to all sources for the event type to create the configuration.

**Dashboard Access**

Grant permission to dashboards in the Dashboard tab. In this tab, the admin specifies if a user can create dashboards and time permissions, as well as custom permissions.

If an Analytics dashboard contains data from business journeys and/or XLM, access to the dashboard depends on the above permissions. For example, a role that allows its users to create a dashboard, but contains no permission to business journeys, does not
permit the user to access a business journey-related dashboard. Ensure that roles have the necessary permission to access Analytics data along with general dashboard access.
# View Analytics Agent Health

You can view 4.5 Analytics Agents connected to your Controller from the AppDynamics Agents page on the Analytics Agents tab. You can view agent status and health information for the Log Analytics and Transaction Analytics pipelines.

When a new Analytics Agent starts and connects to the Controller, it registers with the Controller. Once the agent is registered, the agent requests configuration information every five minutes. As part of this request, the Analytics Agent reports health status information to the Controller. A healthy pipeline means analytics data is flowing from the Analytics Agent to the Events Service successfully.

## Condition

<table>
<thead>
<tr>
<th>Condition</th>
<th>Transaction Pipeline Health</th>
<th>Log Pipeline Health</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analytics Agent does not contact the Controller for more than five minutes.</td>
<td>Not connected 🤔</td>
<td>Not connected 🤔</td>
</tr>
<tr>
<td>No business transaction analytics data is flowing and there are no errors in the connection.</td>
<td>Connection is healthy, no data flowing ✅</td>
<td>Not Applicable</td>
</tr>
<tr>
<td>Business transaction analytics data is flowing and the Agent is publishing to the Events Service successfully with no errors.</td>
<td>Healthy ✅</td>
<td>Not Applicable</td>
</tr>
<tr>
<td>Business transaction analytics data is flowing, but the Agent is experiencing errors while publishing transactions to the Events Service.</td>
<td>Unhealthy 🚨</td>
<td>Not Applicable</td>
</tr>
<tr>
<td>No log analytics data is flowing and there are no errors in the connection.</td>
<td>Not Applicable</td>
<td>Connection is healthy, no data flowing ✅</td>
</tr>
<tr>
<td>Log analytics data is flowing and the Agent is publishing to the Events Service successfully with no errors.</td>
<td>Not Applicable</td>
<td>Healthy ✅</td>
</tr>
<tr>
<td>Log analytics data is flowing, but the Agent is experiencing errors while publishing transactions to the Events Service.</td>
<td>Not Applicable</td>
<td>Unhealthy 🚨</td>
</tr>
</tbody>
</table>

## Permissions

To view the Analytics Agents list, you must belong to a role with the Account level "Administer users, groups, roles, authentication, etc. View License. View AppDynamics Agents. Use Agent Download Wizard" permission. The default Account Owner and Analytics Administrator roles include this required permission.

## Access Analytics Agent Health List

To access Analytics Agents health view, select **Settings > AppDynamics Agents > App Server Agents**.

View information for connected Analytics Agents as follows: unique host id, version, last connected time, the status of log data collection, the status of business transaction data:
Double-click a row to see more agent details:

**Analytics Agent Summary**

- **Name**: analytics-agent1
- **Unique Host ID**: a5433663
- **Version**: 4.4.0
- **Last Connected**: 08/28/17 11:18:09 AM
- **Log Pipeline Health**: ✔
- **Transaction Pipeline Health**: ✔
- **Last Start Time**: 08/29/17 2:31:03 PM
- **Agent Runtime**: 1.8_0.65
- **Install Time**: 08/28/17 11:18:09 AM
- **Install Dir**: /apdynamics/agents/machine-agent/monitors/analytics-agent:

View Analytics Agents by event type (log data and business transaction data).
Perform free text search on the list of registered Analytics Agents.
Application Analytics Licenses

You acquire your Application Analytics license from your AppDynamics sales representative.

There are four types of licenses associated with Application Analytics:

- Transaction Analytics
- Log Analytics
- Browser Analytics
- Mobile Analytics

You do not need an analytics license to view Browser Synthetic Monitoring sessions data. If you have a Synthetic Pro license, you will be able to see the Analytics tab in the controller and if you don't have an Analytics license, the only event type you can see is Synthetic Sessions.

Analytics licenses are based on the following:

- Volume of data:
  - Transaction Analytics: Measured as a specific number of Business Transaction events. Cumulative usage per hour against the daily limit can be viewed on the Administration License page. Do not attempt to compare the number of transactions on the Analytics search page to the number reported in account usage. The analytics search page reports business transactions and the account usage page reports the number of events. There can be several events for each business transaction.
  - Log Analytics: Measured as the size of incoming data. Cumulative usage per hour against the daily limit can be viewed on the Administration License page.
  - Browser: Page views
  - Mobile: Monthly Active Users

- Data retention time may vary depending on your specific license, see License Entitlements and Restrictions

Overages

Analytics does not support overages. AppDynamics stops capturing analytics data after your limit has been reached.

Updating an On-Premises License

If you are updating a previous license, be aware that updated Analytics licenses may not automatically propagate on the Controller across multiple user accounts in an on-premises instance. Updating of these licenses must be managed manually. To update the licenses, open the Administration Console.

1. Select Accounts.
2. Double-click on each account to open.
3. In the top section, adjust the License Date manually.
4. Scroll down to the Application Analytics section, and update the license units provisioned, as necessary.
5. Save.
6. Repeat for all accounts.
Remote Analytics Agent Sizing

On this page:
- Analytics Agent Sizing Based on Event Volume

Related pages:
- Deployment Options and Scenarios
- Installing Agent-Side Components
- Event Service Sizing and Capacity

This topic contains information to help you estimate the hardware requirements for a remote Analytics Agent deployment where a single Analytics Agent is aggregating transaction events from multiple APM Application Agents. This guide covers transaction analytics only.

There are two basic issues regarding sending analytics data to a remote Analytics Agent:
- How many APM agents can report to one remote Analytics Agent?
- What are the machine requirements for hosting the remote Analytics Agent?

Analytics Agent Sizing Based on Event Volume

Based on our testing, the volume of events being sent to the Analytics Agent is the limiting factor in determining how many APM agents can report to one remote Analytics Agent.

The tests were conducted on virtual hardware and programmatically generated workload. Real-world workloads may vary. To best estimate your hardware sizing requirements, carefully consider the traffic patterns in your application and test in a test environment that closely resembles your production application and user activity.

Calculating Analytics Event Volume

One business transaction can traverse many tiers. In each tier, one business transaction traverses one node. One node produces one request per business transaction when the transaction is synchronous. For async transactions, multiple events may be generated by a node for a single request. One request equals one analytics event. To calculate how many events a business transaction generates, you need to count the number of tiers/nodes that are sending data into the Analytics Agent.

You can estimate the number of events using the following formula:

One business transaction generates events at a rate = \text{calls per minute} \times \text{the number of tiers} reporting analytics data for the business transaction.

In simple terms: \#events for one business transaction = \text{calls per minute} \times \# of tiers.

Characteristics of the Amazon EC2 Instance Types

For complete information on Amazon EC2 instance types, see https://aws.amazon.com/ec2-instance-types/.

The testing was performed using c3.large, c3.xlarge, and c4.4xlarge.

<table>
<thead>
<tr>
<th>Model</th>
<th>vCPU</th>
<th>Mem (GiB)</th>
<th>SSD Storage (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>c3.large</td>
<td>2</td>
<td>3.75</td>
<td>2 x 16</td>
</tr>
<tr>
<td>c3.xlarge</td>
<td>4</td>
<td>7.5</td>
<td>2 x 40</td>
</tr>
</tbody>
</table>
Testing Results

**c3.large CPU% vs. AA events/min**

**c3.xlarge CPU% vs. AA events/min**
Raw Data

<table>
<thead>
<tr>
<th>Analytics Agent Host Machine</th>
<th>Analytics Agent events/min</th>
<th>Total CPU%</th>
<th>JVM Heap (Mb)</th>
</tr>
</thead>
<tbody>
<tr>
<td>c3.large</td>
<td>52313</td>
<td>17%</td>
<td>468</td>
</tr>
<tr>
<td>c3.large</td>
<td>54831</td>
<td>17%</td>
<td>470</td>
</tr>
<tr>
<td>c3.large</td>
<td>70746</td>
<td>24%</td>
<td>475</td>
</tr>
<tr>
<td>c3.large</td>
<td>74541</td>
<td>24%</td>
<td>477</td>
</tr>
<tr>
<td>c3.large</td>
<td>77344</td>
<td>23%</td>
<td>487</td>
</tr>
<tr>
<td>c3.large</td>
<td>97074</td>
<td>28%</td>
<td>512</td>
</tr>
<tr>
<td>c3.large</td>
<td>115999</td>
<td>36%</td>
<td>519</td>
</tr>
<tr>
<td>c3.large</td>
<td>139143</td>
<td>43%</td>
<td>526</td>
</tr>
<tr>
<td>c3.large</td>
<td>148782</td>
<td>47%</td>
<td>587</td>
</tr>
<tr>
<td>c3.large</td>
<td>204073</td>
<td>65%</td>
<td>527</td>
</tr>
<tr>
<td>c3.large</td>
<td>247543</td>
<td>80%</td>
<td>624</td>
</tr>
<tr>
<td>c3.large</td>
<td>249261</td>
<td>81%</td>
<td>637</td>
</tr>
<tr>
<td>c3.xlarge</td>
<td>196288</td>
<td>33%</td>
<td>518</td>
</tr>
<tr>
<td>c3.xlarge</td>
<td>254586</td>
<td>44%</td>
<td>727</td>
</tr>
<tr>
<td>c3.xlarge</td>
<td>302689</td>
<td>51%</td>
<td>497</td>
</tr>
<tr>
<td>c3.xlarge</td>
<td>336879</td>
<td>58%</td>
<td>913</td>
</tr>
<tr>
<td>c3.xlarge</td>
<td>372515</td>
<td>65%</td>
<td>1024</td>
</tr>
<tr>
<td>c3.xlarge</td>
<td>513598</td>
<td>91%</td>
<td>922</td>
</tr>
<tr>
<td>c3.xlarge</td>
<td>478954</td>
<td>78%</td>
<td>922</td>
</tr>
<tr>
<td>c3.xlarge</td>
<td>420000</td>
<td>69%</td>
<td>979</td>
</tr>
<tr>
<td>c3.xlarge</td>
<td>376034</td>
<td>61%</td>
<td>1024</td>
</tr>
<tr>
<td>c3.xlarge</td>
<td>318000</td>
<td>52%</td>
<td>1024</td>
</tr>
<tr>
<td>Instance Type</td>
<td>CPU Usage</td>
<td>Memory</td>
<td>% CPU</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------</td>
<td>--------</td>
<td>-------</td>
</tr>
<tr>
<td>c3.xlarge</td>
<td>258000</td>
<td>1024</td>
<td>43%</td>
</tr>
<tr>
<td>c3.xlarge</td>
<td>198000</td>
<td>1024</td>
<td>32%</td>
</tr>
<tr>
<td>c3.xlarge</td>
<td>144000</td>
<td>1024</td>
<td>22%</td>
</tr>
<tr>
<td>c4.4xlarge</td>
<td>534900</td>
<td>552</td>
<td>17%</td>
</tr>
<tr>
<td>c4.4xlarge</td>
<td>604725</td>
<td>841</td>
<td>19%</td>
</tr>
<tr>
<td>c4.4xlarge</td>
<td>716141</td>
<td>1024</td>
<td>23%</td>
</tr>
</tbody>
</table>
Enable SSL for the Analytics Agent

Related pages:
- Enable SSL for the Java Agent
- Controller SSL and Certificates

This topic describes how to configure the AppDynamics Analytics Agent to connect to your app server agents using SSL.

1. Run the keytool command to create a new key pair for the Analytics Agent in the keystore. Refer to the Oracle documentation for more details on using keytool. The following command creates a keystore if it doesn't exist and generates the public/private key pair:

```bash
keytool -genkeypair -alias analytics-agent -keystore aa-keystore.jks -validity 1825
```

Follow the on-screen instructions to configure the certificate. This generates a self-signed certificate in the keystore. The next step generates a signing request for the certificate. Note the following:

a. For the first and last name, enter the domain name where the Analytics Agent is running. The domain name used in the `appdynamics.analytics.agent.url` property must match the Common Name (CN) of the certificate used by the Analytics Agent. For example, if the URL was `https://localhost:9090/v2/api` then the CN of the certificate should be `localhost`. If the names don't match then the client marks the certificate as invalid and the HTTPS connection is not established.

b. Enter a secure password for the key. This command creates a key pair with a validity of 1825 days (5 years). Replace 1825 with the validity period appropriate for your environment.

2. Generate a certificate signing request for the certificate you created as follows:

```bash
keytool -certreq -alias analytics-agent -keystore aa-keystore.jks -file AppDynamics.csr
```

3. Submit the certificate signing request file generated by the command (`AppDynamics.csr` in the example command) to your Certificate Authority (CA) of choice.

4. When it's ready, the CA returns the signed certificate and any root and intermediary certificates required for the trust chain. The response from the CA should include any special instructions for importing the certificate if needed. If the CA supplies the certificate in text format, copy and paste the text into a text file.

5. Import the signed certificate:

```bash
keytool -import -trustcacerts -alias analytics-agent -file mycert.cer -keystore aa-keystore.jks
```

This command assumes the certificate is located in a file named `mycert.cer`.

6. If you see the error “Failed to establish chain from reply”, install the issuing CA's root and any intermediate certificates into the keystore. The root CA chain establishes the validity of the CA signature on your certificate. Although most common root CA
chains are included in the bundled JVM’s trust store, you may need to import additional root certificates, such as certificates belonging to a private CA. To do so:

```
keytool -import -alias [Any_alias] -file
<path_to_root_or_intermediate_cert> -keystore
<controller_home>/appserver/glassfish/domains/domain1/config/aa-keystore.jks
```

7. When done importing the certificate chain, try importing the signed certificate again.
8. Update the following properties in the analytics-agent.properties file:
   a. ad.dw.https.enabled=true to enable the HTTPS connector on the Analytics Agent. The HTTPS connection is exposed on the port defined by the ad.dw.http.port property.
   b. ad.dw.https.keyStorePath= absolute path to the Java keystore that contains the Analytics Agent public and private key.
   c. ad.dw.https.keyStorePassword= Java keystore password.
   d. ad.dw.https.trustStorePath= absolute path to the truststore that establishes the chain of trust for the Analytics Agent public key certificate.
   e. ad.dw.https.trustStorePassword= the truststore password.
   f. ad.dw.https.certAlias= alias of the public key certificate stored in the Java Key Store.
9. Start the Analytics Agent.
   The HTTPS connection should now be exposed.
10. Confirm that the app server agent is configured to trust the Analytics Agent certificate. See the section “Enable SSL between the Java Agent and the Analytics Agent” in the Enable SSL for the Java Agent topic.
Troubleshooting Analytics Agent Issues

On this page:
- General Troubleshooting
- Configuration Issues
- Clock Management and Timestamps
- Understanding Analytics Data and Timestamps
- Limits for Business Transaction Events
- Issues Starting the Analytics Agent
- Log Analytics Missing Field Extractions
- Custom Analytics Metrics
- Firewall Considerations
- Monitoring Health of the Analytics Agent
- Known Issues

Related Pages:
- Analytics Agent Logging
- Agent Log Files

This topic describes general techniques when troubleshooting AppDynamics Analytics deployment problems, along with a few specific scenarios you may encounter with workarounds.

General Troubleshooting

If you encounter problems with your Analytics deployment, first check the logs for errors or warnings. The components write log information as follows:

- Analytics Dynamic Service: The Analytics Dynamic Service is built into the Java and .NET App Agents and writes logs to the same file as the App Agent. The logs are in the following location:
  `<application_home>/app_agent_home/logs`.
  The primary log file to use for troubleshooting is the file named `agent.<timestamp>.log`. Search the file for messages written by the Analytics Dynamic Service. These logs are useful for the following:
  - Determining if the Analytics Dynamic Service is enabled or disabled.
  - Viewing the configurations used on startup.
  - Determining if the Analytics Dynamic Service is encountering errors sending messages to the Analytics Agent. For example, when the Analytics Dynamic Service is not able to communicate with the Analytics Agent due to invalid connection configuration.
  - Determining if messages are being dropped by the Analytics Dynamic Service because its internal buffers are full.

- The Analytics Agent writes log messages to files in the following directory:
  `<analytics_agent_home>/logs`

- The Events Service writes log messages to files in the following directory:
  `<events_service_home>/logs`
  In particular, the `analytics-api-store.log` file can help you with troubleshooting.

Configuration Issues

Verify your configuration settings, particularly that they are properly configured with the required account name and key. Slashes in account names and key values need to be escaped, with the failure to do so a common source of configuration problems.

Clock Management and Timestamps

AppDynamics recommends maintaining clock-time consistency throughout your monitored environment. If analytics metrics are always reporting zero, confirm that the clocks are synchronized across the application, Controller, and Events Service nodes.

Understanding Analytics Data and Timestamps
There are potentially four time zones involved when dealing with log analytics, which you should be aware of:

1. The timestamp and time zone from the log file.
2. The event timestamp (and pickup timestamp) time zones can be different from that in the log for a number of reasons, such as the following:
   a. When the time zone is overridden
   b. The time zone is not provided correctly in the log
   c. The timestamp and time zone parsing goes awry
   d. When no time zone is specified in the log timestamp, then local time is assumed
3. The Events Service time zone, the Events Service stores all timestamps in UTC time.
4. The browser used to view the analytics data in the Controller UI (such as event timestamp column displayed in the UI search results or the time picker widget) converts all timestamps to the browser’s local time.

Limits for Business Transaction Events

The Analytics Dynamic Service sends messages to the Events Service where the request body is an array of event segments. A business transaction event consists of one or more segments that are related to each other by the business transaction requestGUID.

There are ingestion limits related to messages:

- Event (segment) size: The maximum size of an individual business transaction segment collected by the Analytics Dynamic Service is .1 MB. This limit is defined by the appdynamics.analytics.message.maxSizeBytes Java system property. To change this value, pass it on the command line as a system property when the Java Application Agent is started, for example:

```
-Dappdynamics.analytics.message.maxSizeBytes=1024000
```

- Events per request: The maximum number of segments per request is defined by the appdynamics.analytics.agent.send.batch.items.max Java system property. The default value of this property is 16. To change this value, pass it on the command line as a system property when the Java Application Agent is started.

- Message Size: This limit refers to the size of a single request body sent to the Events Service, which is usually an array of event segments. Publish requests for all event types are limited to 1 MB. If the limit is exceeded, you will see exceptions in agent log file and messages in Events Service logs.

Issues Starting the Analytics Agent

If an instance of the Analytics Agent terminates and leaves behind its process ID file (PID file), the next agent startup will fail with the following error:

```
java.lang.RuntimeException: Unable to create file [D:\AppDynamics\analytics-agent\analytics-agent.id] to store the process id because it already exists. Please stop any currently running process and delete the process id file
```

In Controller versions prior to 4.3, you needed to delete the old process id file and restart the agent to work around this issue. In 4.3 and higher, you can use the –f option when starting the agent. This option causes a preexisting process id file to be deleted. The flag is not required when you are starting the agent as a Windows service.

Use the –f flag as follows:

- UNIX type OS: `start -f`
- Windows CLI: `start -f`
Log Analytics Missing Field Extractions

If you are missing fields in your Log Analytics data that you expect to see based on your source rule configuration, if you are using regex (including grok patterns) in your field extraction, you may be encountering a performance safeguard.

If a regex pattern takes more than five seconds to match against a log line, the attempt to extract the fields is terminated, and no further processing occurs requiring the extracted fields. As a result, some fields may be missing when viewed on the controller for that log line. In this case, the following error message appears in the analytics agent log:

```
[ERROR ] java.util.concurrent.TimeoutException: The current regex has spent 5 seconds attempting to match the log line, further processing has been stopped for this log line.
```

Another reason for missing fields is if the log line doesn't contain the field to be extracted as defined in the pattern.

Custom Analytics Metrics

If you are having issues with metrics created from saved searches or with the alerts performance for those metrics, try increasing the query batch size. You can increase the size using the `analytics.scheduledqueries.batch.size` Controller setting in the Controller Administration Console. The default value for this setting is 5.

See Access the Administration Console for information about accessing the setting.

Firewall Considerations

For SaaS-based installations, you configure the Analytics endpoint by modifying the `http.event.endpoint` setting in the `conf/analytics-agent.properties` file (as described in Installing Agent-Side Components). For example:

- `http.event.endpoint=http://analytics.api.appdynamics.com:443`

If your firewall rules require you to use specific IP addresses, rather than hostnames, note the following information. If you are unable to see transaction analytics data collected as expected (even after configuring your firewall rules) and you see repetitive "Connection Reset" messages in the logs similar to the following, your firewall rules may not include the correct IP addresses.
Your firewall rules may not include the correct IP addresses.

In SaaS environments, both analytics.api.appdynamics.com, fra-ana-api.saas.appdynamics.com and syd-ana-api.saas.appdynamics.com are round-robin DNS aliases and may resolve to multiple DNS (54. vs 52.) such as in the following examples:

<table>
<thead>
<tr>
<th>Name: analytics.api.appdynamics.com</th>
<th>Name: fra-p-con-2.saas.appdynamics.com</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address: 54.213.173.141</td>
<td>Address: 52.28.42.67</td>
</tr>
<tr>
<td>Name: analytics.api.appdynamics.com</td>
<td>Name: fra-p-con-2.saas.appdynamics.com</td>
</tr>
<tr>
<td>Address: 52.88.111.157</td>
<td>Address: 52.57.96.225</td>
</tr>
</tbody>
</table>

Amazon Web Services (AWS) controls the IPs used, so they may change from time to time. AWS publishes its current IP address ranges in JSON format, so if you are unable to open firewalls to hostnames, you can download the AWS IP address ranges. If you want to be notified whenever there is a change to the AWS IP address ranges, you can subscribe to receive notifications using Amazon SNS.

See Analytics IP Ranges for the AWS regions for the SaaS Analytics environments. You can view the AWS IP ranges for each listed region.

**Monitoring Health of the Analytics Agent**

The check-health command returns the status of an analytics agent. You can specify the agent to check using a properties file or IP address and port. For example, using a properties file:

```
./bin/analytics-agent.sh check-health --properties conf/analytics-agent.properties
```

You can also use the `-hp` argument and pass the host IP address and the port number for the analytics agent. For example:

```
bin/analytics-agent.sh check-health -hp <ip-address>:9091
```

The default analytics agent port for the health check is 9091.

Usage for the check-health command is:
check-health
- hp (--host-and-port) STRING[]
  (Optional. Application host name or IP:HTTP admin port (Multiple values separated by space))
- p (--properties) TYPE_PATH
  (Optional. Path to the properties file (Multiple values separated by space))
- v (--verbose) TYPE_BOOLEAN
  (Optional. Verbose output)

Known Issues

- On Windows, you cannot delete a log file with the "del" command while the analytics agent is collecting log data from the file.
- Avoid using robocopy/move commands to move files in Windows. Instead, it is recommended that you use the "move" command.
Analytics Agent Logging

By default, the AppDynamics Analytics Agent writes log files to the `<analytics_agent_home>/logs` directory. See Agent Log Files for information about how the logs are organized into sets that roll over.

Log File Sizes

Each log can grow to 64 MB in size before being compressed and renamed. A maximum of 10 compressed files can be generated.

The default pattern for agent log naming is `<filename>-<date_compressed>.log.gz` as shown in the table below.

<table>
<thead>
<tr>
<th>log file name</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>analytics-agent.log</td>
<td>the current log</td>
</tr>
<tr>
<td>analytics-agent-2018-12-11.log.gz</td>
<td>the previous log</td>
</tr>
<tr>
<td>analytics-agent-2018-12-10.log.gz.gz</td>
<td>the 2nd previous log</td>
</tr>
</tbody>
</table>

Logging Level for the Analytics Agent

The default logging level for most log files is INFO. Higher logging levels consume more disk space.

You can control the logging level for the Analytics Agent by changing the value of the `ad.dw.log.level` parameter in the `analytics-agent.properties` file.

For example, to set the log level to DEBUG:

```
ad.dw.log.level=DEBUG
```
Configuring Application Analytics

Once you have installed the components and enabled Application Analytics, you need to configure the system to collect the data in which you are interested.

1. In the Controller UI, from the top navigation bar, click Analytics.
2. In the left navigation panel, click Configuration.
3. From the Transaction Analytics tab, enable analytics for specific applications and business transactions. You can also enable analytics for existing data collectors or create new ones specifically for your analytics needs.
4. From the API Keys tab, create API authentication keys for users of the Analytics Events API.
5. From the Log Analytics tab, you can configure source rules to collect data for log analytics.

Browser Request Analytics and Mobile Request Analytics, alternate views of EUM data, do not require configuration. If you are using an on-premises Events Service, however, and an on-premises EUM Server, you do need to configure the Server to send its data to the Events Service. See EUM Server Deployment.

Related pages:
- Business Transaction and Log Correlation
- Analytics Events API
Collect Transaction Analytics Data

On this page:
- Select the Application and the Business Transactions
- Configure Data Collectors
- Enable Transaction Analytics Data Type Conversion for Data Collectors
- Configure Analytics for Node.js
- Configuring Analytics for PHP Agent

Related pages:
- Configuring Application Analytics
- Business Transaction and Log Correlation
- Data Collectors

Works with:

Java .net .NET

Make sure you have installed and configured the components described in Installing Agent-Side Components and, for on-premises, Custom Install and Events Service Deployment before attempting to configure Transaction Analytics.

Select the Application and the Business Transactions

Configuring Transaction Analytics consists of selecting the application and the specific business transactions that you want to analyze. You can also enable the collection of additional business data using Data Collectors. For Node.js apps, see Configure Analytics for Node.js.

To enable and configure Transaction Analytics for Java and .NET applications:

1. In the Controller UI, from the top navigation bar, select Analytics > Configuration.
2. From the Transaction Analytics tab, select the application.
3. Check the checkbox labeled Enable Analytics Data Collection for <selected_app>.
4. Specify which business transactions should report analytics data by adding available transactions from the right-hand list to the left-hand list of transaction reporting analytics data:
Even if you have pre-existing Data Collectors defined, you do not get Analytics data unless you confirm that the related business transactions are enabled for analytics.

The Analytics Dynamic Service loads Transaction Analytics for enabled applications after your JVM instance runs for at least 2 minutes.

Configure Data Collectors

After you select the application and business transactions, you can optionally configure data collectors. The Configure Transaction Analytics page includes sections you can use to configure HTTP and method invocation data collectors, as follows.

HTTP Data Collectors

To collect HTTP request data, you can use the default HTTP Request Data Collector. You need to explicitly enable the collector for Analytics.

To configure HTTP Data Collectors for Analytics, use these steps:

1. In the Controller UI, select Analytics > Configuration from the top navigation bar.
2. From the Transaction Analytics tab, select the application of interest and scroll to the HTTP Request Data Collector section.
3. Select the HTTP Data Collector to enable and click Edit (or Add to create new collectors).

4. On the HTTP Request Data Collector page, confirm the data to collect, confirm Transaction Analytics is checked, and click Save.
4. Use **Configure Transactions Using this Data Collector** and confirm that the data collector is enabled on the appropriate business transactions. See **Data Collectors** for more information.

5. **Method Invocation Data Collectors**

You can also use the **Analytics > Configuration** page to enable existing or new Method Invocation Data Collectors. Open the **Method Invocation Data Collectors** panel. The process is essentially the same as described in **Data Collectors**.

Make sure that you:

- Check the **Transaction Analytics** checkbox to use this collector for Application Analytics
Check the Configure Transactions popup to confirm you have enabled the right Business Transactions.
Check the name of the MIDC does not exceed the 50 character limit.

Changing the name of method invocation data collector will create a completely new field in transaction analytics. If you want to change the type of a data collector field, we recommend creating a new field instead to avoid conflicts with the old data.

SQL Data Collectors
Transaction snapshots capture SQL queries. SQL data collectors provide a way to extract business data from parameters used in the SQL statements for use in analytics. See Collect Business Data From SQL Calls.

Enable Transaction Analytics Data Type Conversion for Data Collectors

Transaction analytics data type conversion requires controller 4.5.6+
For Java deployments, transaction analytics data type conversion is available for Java agents version 4.5.2+

Data collectors determine field type (string, boolean, or number) at runtime. At times the data you collect may not be the type in which you want to use it. For example, you are not able to add metrics if any data collectors are field type string.

While field type cannot be changed, you can specify field type before runtime with transaction analytics data type conversion.

To enable transaction analytics data type conversion:

1. Log in to the administration console:
   http://<controller-hostname>:8080/controller/admin.jsp or
   https://<controller-hostname>:45/controller/admin.jsp
2. On the Controller Settings page, change the value of analytics.type.conversion.enabled from false to true.
3. Press Save.

Data type conversion only appears when enabled in HTTP and Method Invocation Data Collector configuration pages on your controller. If this page was opened while you enabled data type conversion, you will need to refresh your page to see changes.

Data type conversion operates by the following rules:

- Available for new data collectors only
- Select type manually for each data collector
- Type cannot be changed once saved

Transaction Analytics Data Type Conversion in HTTP Data Collectors

To use transaction analytics data type conversion, create a new HTTP parameter. You can add to an existing HTTP Request Data Collector or create a new one.

The process is essentially the same as creating an HTTP parameter. However, now that you've enabled transaction analytics data type conversion, you will see “Type” field with a drop-down menu, as seen in the screenshot below:

<table>
<thead>
<tr>
<th>HTTP Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="HTTP Parameters Table" /></td>
</tr>
</tbody>
</table>

Select the desired field type for your HTTP parameter. If you do not want to specify a type, leave this field blank. The default type for
HTTP data collectors is string.

**Transaction Analytics Data Type Conversion in Method Invocation Data Collectors**

To use transaction analytics data type conversion, create a new Method Invocation Data Collector (MIDC). You can add to an existing MIDC or create a new one.

The process is essentially the same as creating an MIDC. However, now that you've enabled transaction analytics data type conversion, you will see "Change Type to" field with a drop-down menu, as seen in the screenshot below:

Select the desired field type for your MIDC.

Transaction Analytics Type defaults to "Use APM Type," which determines type at runtime. If you do not want to specify a type, select "Use APM Type."

**Naming requirements for transaction analytics data type conversion**

Refer to the following naming requirements when using transaction analytics data type conversion:

- Display Names must be unique within each data collector. Names are case sensitive.
- You can repeat Display Names across different data collector configurations, as long the field type is also the same.
- If you delete a field from a data collector and add it again with a different field type, you must rename the data collector.

**Changing field types**

Field types cannot be changed once they have been saved to the data collector. To change the field type, we recommend to delete and recreate with a new name.

**Configure Analytics for Node.js**

**Collect Default Node.js Transaction Data**

To configure the Node.js Agent to send the default transaction data to the analytics agent, modify the require statement in your application. Add the following variables with the proper values to point to your Analytics Agent:
The Analytics Agent can be on the same host as the Node.js Agent, or on a different host. For more details on installing the Node.js Agent, see Install the Node.js Agent.

For example, your require statement should look similar to the following where your Analytics Agent is on localhost and listening on port 9090:

```javascript
require("appdynamics").profile({
  controllerHostName: '<controller host name>',
  controllerPort: <controller port number>,
  controllerSslEnabled: false, // Set to true if controllerPort is SSL
  accountName: '<AppDynamics_account_name>',
  accountAccessKey: '<AppDynamics_account_key>', //required
  applicationName: 'your_app_name',
  tierName: 'choose_a_tier_name',
  nodeName: 'choose_a_node_name',
  analyticsHostName: 'host_name',
  analyticsPort: 'port_number',
  analyticsSslEnabled: false, // Set to true if controllerPort is SSL
  analytics: { // Set to true if controllerPort is SSL
    host: 'localhost',
    port: 9090
  }
});
```

**Node.js Data Collectors**

You can also collect additional data from your Node.js business transactions by using the Node.js API. To configure transaction analytics data collectors for Node.js, see `txn.addAnalyticsData()` in the Node.js Agent API Reference.

**Configuring Analytics for PHP Agent**

1. To configure the PHP Agent to send the default transaction data to the analytics agent, modify the .ini file, depending on the operating system under which your PHP is installed. Add the following variables with the proper values to point to your Analytics Agent:

```ini
agent.analyticsHostName = <analyticsHostName>
agent.analyticsPort = <analyticsPort>
```
The Analytics Agent can be on the same host as the PHP Agent, or on a different host. For more details on installing the PHP Agent, see Install the PHP Agent.

For example, your .ini file should contain the following where your Analytics Agent is on localhost and is listening on port 9090:

```
agent.analyticsHostName = localhost
agent.analyticsPort = 9090
```

2. HTTP/Method Invocation Data collectors for analytics must be enabled for both Transaction snapshots and Transaction Analytics for data to be reported from the PHP agent.

3. In order to collect method return value with an MIDC, the value must be assigned to a variable.

   ```php
   $ret = function()
   ```

   If the return value is not stored in any variable then it will be seen as null in both snapshot and Analytics data.

4. Analytics data cannot currently be collected from PHP CLI programs.
Add Custom Fields to Transactions Using Java SDK

You can add custom data using the Java Agent SDK as an alternative to adding data collectors from the Controller UI. The fields show up on Transaction Analytics under a section called "Custom Method Data".

Add the code similar the following to your application:
//include the app agent SDK packages

import com.appdynamics.apm.appagent.api.AgentDelegate;
import com.appdynamics.apm.appagent.api.DataScope;
import com.appdynamics.apm.appagent.api.IMetricAndEventReporter;

public void someMethod(String stringParam, int intParam, ComplexObject complexObjParam) {

    //Get the data reporter.
    IMetricAndEventReporter reporter = AgentDelegate.getMetricAndEventPublisher();

    //Define the scope for data collection.
    Set<DataScope> allScopes = new HashSet<DataScope>();
    allScopes.add(DataScope.ANALYTICS);
    allScopes.add(DataScope.SNAPSHOTS);

    Set<DataScope> analyticsScope = new HashSet<DataScope>();
    analyticsScope.add(DataScope.ANALYTICS);

    //Add data to different scopes based on the need.
    //Here both key1 and key2 are collected for both snapshots and analytics
    reporter.addSnapshotData("key1", stringParam, allScopes);
    reporter.addSnapshotData("key2", intParam, allScopes);

    //Here key3 is only collected for analytics.
    //Since the passed object is a complex java object and not a primitive type, toString() method is invoked on it and the string value will
    //be collected.
    reporter.addSnapshotData("key3", complexObjParam, analyticsScope);

    ........
    Business logic.

The third parameter to the method is a set of enums, DataScope, that define the scope for collecting the custom data:

- To collect custom data only for snapshots, use DataScope.SNAPSHOTS as the third parameter.
- To collect the custom data for analytics, use DataScope.ANALYTICS as the third parameter.
- If custom data should be added to both snapshots and analytics then both the enums should be passed in a set as the third parameter.

Method details for addSnapshotData()
boolean addSnapshotData(java.lang.String key,
                java.lang.Object value,
                java.util.Set<DataScopes> dataScopes

Method Parameters

- **key** - key
- **value** - Primitive types can be passed as values and the types (int, boolean, etc) will be respected. Strings can be passed as values, however, note that if you pass composite objects then agent performs a .toString() operation on it.
- **dataScopes** - scope for the custom data collection:
  - DataScope.SNAPSHOTS
  - DataScope.ANALYTICS

Returns

true if the agent successfully collects the key-value pair for the given scope
Collect Business Data From SQL Calls

Transaction snapshots capture SQL database calls. The SQL calls can contain useful business data. Analytics SQL data collectors are a way to collect the business data from SQL parameters for use in transaction analytics.

To configure an Analytics SQL data collector you need to know the following:

- Analytics-enabled application where the SQL call is executed.
- Database target of the SQL call.
- Specific SQL parameterized statement that contains the data of interest as a query parameter.
- Analytics-enabled business transactions making the database call.
- Parameters to be collected.

This feature is supported as follows:

- The SQL data collector list shows the slowest database calls over the last 30 days. You can also create a SQL data collector directly from the tab of a snapshot.
- Requires 4.3 Java Agent or 4.3 .NET Agent.
- Only prepared statements containing data of interest as binding variables can be used. Literal strings that are passed in can not be collected.
- There is a 500-character limit on the length of the SQL statement. Do not use any truncated queries that might appear in the Statements list or a snapshot.
- The overall number of executions of SQL queries configured to collect Analytics data is limited to 10K. This is configurable using the `analytics-sql-cpm-limit` node property. See App Agent Node Properties.

Configure SQL Data Collectors From Analytics

1. In the Controller UI, from the top navigation bar, select Analytics > Configuration.
2. From the Transaction Analytics tab, select the application from the Configure Analytics for Application drop-down list and confirm that analytics data collection is enabled.
3. Scroll down to expand the SQL Data Collectors section, and click Add.

4. Name your data collector and indicate if the collector should apply to new business transactions.
5. Select the appropriate database.

Once you do this, a list of available SQL statements is displayed showing the slowest database calls over the last 30 days. This timeframe is not configurable.
6. Define the data to collect.
   a. Select the SQL prepared statement containing the parameter that you want to capture for analytics.
   b. Click Add to specify the data to be collected.
   c. Type a display name for the data you are collecting. This name appears in the Analytics UI Fields list when the data is collected and passed to Analytics.
   d. Specify the data type and method parameter index.
   e. Click Save.
   f. Click Create SQL Data Collector.

7. Specify the business transactions that will use this collector and click Save.
Configure SQL Data Collectors From Snapshots

This procedure provides a shortcut way to set up the configuration for an Analytics SQL data collector for a database call captured in an application transaction snapshot. You need to find the transaction snapshot of interest and drill down to the node that contains the database call of interest.

1. From the **DB & Remote Service Calls** tab of a snapshot containing the SQL call of interest, select the prepared statement that contains the data you want to collect for analytics.
2. Right-click the query and select **Configure Data Collector** from the context menu. You see the Data Collection window.
3. Type a display name for the data you are collecting.
4. Specify the type and the parameter to collect and click **Save**.
5. Select the business transactions from which to collect the data and click **Save**.

**View SQL Data in Analytics**

After the data is successfully collected for analytics, you see the fields in the Fields list in a section labeled SQL Data. This can take a few minutes depending on your application and the frequency of transactions executing the relevant SQL query.
Collect Log Analytics Data

To capture and present log records as analytics data, you must configure one or more log sources for the Analytics Agent. The Analytics Agent uses the log source configuration to:

- Capture records from the log file
- Structure the log data according to your configuration
- Send the data to the Analytics Processor.

The Controller presents the Log Analytics data in the Application Analytics UI.

Before attempting to configure Log Analytics, confirm you have installed and configured the components described in Installing Agent-Side Components and, for on-premises, Custom Install and Events Service Deployment.

Versions prior to 4.3 use job files to configure the log sources. You may continue to use job files that were created in previous versions. If you want to collect new log events into our platform, we recommend that you use the Centralized Log Management UI to define source rules. You may also find it useful to replace existing job file configurations with the new source rules so you can take advantage of new features introduced in 4.3. See Migrate Log Analytics Job Files to Source Rules for details.

To configure data collection for your log sources, see Configure Log Analytics Using Source Rules.

Manage Extracted Fields

In Controller versions prior to 4.3, you could extract fields from logs in the Controller UI. This is described at Create Extracted Fields from Logs in the 4.2 documentation. This option is not available in versions 4.3 and higher. It is replaced entirely by Field Extraction in the Centralized Log Management UI. See Configure Log Analytics Using Source Rules for details.

Fields that were extracted by this mechanism in previous versions appear in the Extracted Fields list. Hovering over the field reveals a View icon. Click the View icon to delete the field or to see the configuration details:

- Regular expression used to extract the field
- Field name
- Field type
- Source type
Configure Log Analytics Using Source Rules

The Centralized Log Management UI enables you to configure your log data sources using source rules.

Once you define source rules, you specify which Analytics Agents should use the rules by associating specific source rules with Agent Scopes. Agent Scopes are groups of Analytics Agents that you define. Using Agent Scopes simplifies the deployment of the log source configuration to multiple agents.

About Log Analytics Source Rules

Analytics Agents collect logging information based on the log source rules you define in the Controller.

The primary function of a source rule is to specify the location and type of a log file, the pattern for capturing records from the log file, and the structure of the data of captured records. They can also specify field masking or sensitive data removal and manage time zones of captured records.

When the log source rules are enabled and associated with Agent Scopes, Analytics Agents automatically start collecting the configured logs as follows:

- Analytics Agents register with the Controller on startup
- Analytics Agents download log source rules to configure log collection (after registration)
- Log source rules are stored in the Controller data store and are configurable through the Centralized Log Management UI
- Analytics Agents start acting on log source rule changes within five minutes (this could be longer if there are any network communication issues)

Permissions

To configure log analytics source rules, you need the following permissions:

- Manage Centralized Log Config permission
- View access to the specific source types that you want to configure

For more details, see Analytics and Data Security.

Analytics Agent Properties

If you are using Centralized Log Management (CLM), you need to configure additional properties in the analytics-agent.properties file. See information on configuring properties for consolidated log management in Installing Agent-Side Components for details.

Managing Source Rules

When you create a source rule and save it, the rule is initially disabled. This is shown in the Status column on the source rule list page.

You can enable or disable a source rule by selecting it and choosing the corresponding option from the Source Rules menu, shown in the following screenshot:
You can also see which source rules use a specific agent scope. The Analytics Agent does not collect the log data until the source rule is enabled and assigned to an Agent Scope with active Analytics Agents.

Overview of Creating a Source Rule

You can create a source rule from any one of the following starting points:

- AppDynamics template. Several templates for common log file formats are available.
- Existing source rule. You can use existing source rules as the starting point for new rules.
- New source rule. Start from scratch when your log file does not match one of the available templates.

Preview Extracted Log Data Using Sample Log Files

To improve validation of data collection and parsing of log messages, you can use a local log file with the Log Analytics Configuration UI to preview the field extraction that you want. Three types of field extraction are available.

- Grok patterns and key-value pair extraction
- Auto extraction using regular expressions
- Manual extraction using regular expressions

Best Practices for Source Rule Design

A few recommendations apply to making source rules.

Use the simplest possible regular expressions and grok matching patterns possible. Do not make excessive use of wildcards or quantifiers as it can slow the responsiveness of the Controller UI. You can view examples of such greedy quantifiers on https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html.

If a regex pattern (including grok) is taking more than five seconds to match against a logline, extraction and further processing for those fields stop. If this occurs, some fields may be missing for that log line when viewed on the Controller. Other log lines are not impacted; however, this occurrence is often the result of an ineffective or faulty matching pattern in the first place, and processing is likely to take a long time for all log lines. This behavior is applicable to the dynamic Preview screens in the Centralized Log Analytics Configuration page as well.

The Controller limits the size of the records it retrieves to 32 KB by source rule in log analytics. This limit guards against excessive system resource burden, including potential resource burden that might be caused by excessive data collection due to faulty source rule patterns.

Create a Source Rule

1. From the Controller top navigation bar, click Analytics.
2. From the left navigation panel, click Configuration > Log Analytics.
   You see two tabs, one for Source Rules and one for Agent Scopes.
3. From the Source Rules tab, click + Add.
   You see the Add Source Rule window.
4. In the Add Source Rule window, select your starting point for the source rule. Note that you can also use a job file as the starting point for your source rule. Also, see Migrate Log Analytics Job Files to Source Rules.
5. Use **Collection Type** to indicate if the source log file resides on the local filesystem or will be collected from a network connection.

Collecting from a network connection is only used for a TCP source rule, which extracts log analytics fields from \((\text{syslog})\) messages over TCP. See Collect Log Analytics Data from Syslog Messages for details.

6. Use the **Browser** button to locate and specify a sample log file to preview the results of your configuration. You can also specify a sample file later in the configuration process during the field extraction step.

7. Click **Next** to see the Add Source Configuration page. Some fields may be prepopulated with data if you selected either an
AppDynamics template or an existing source rule as your starting point. The four tabs are:

- General
- Field Extraction
- Field Customization
- Agent Mapping

8. On the General tab, name your rule, specify its location, timestamp handling, and other general characteristics. For more details, see General Configuration.

9. On the Field Extraction tab, configure the fields that you want to capture from your log file. For more details on the fields on this subtab, see Field Extraction. For a detailed procedure on using Auto Field Extraction, see Field Extraction for Source Rules.

Use the simplest possible regular expressions and grok matching patterns. Do not make excessive use of wildcards or quantifiers as it can slow the responsiveness of the Controller UI. Examples of expensive and greedy quantifiers can be found here: https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html

10. On the Field Management tab, you can customize the handling of fields in a number of way including masking sensitive data, renaming fields, changing the data type and more. For more details, see Field Management.

Mask Value: Use this option to mask values of sensitive data such as credit card numbers or social security numbers. Select Mask value and enter the starting index and ending index (1 and 4 respectively for the example shown) and a character to use for masking in the display, such as an * asterisk.

Replace Value: Use this option to replace the entire data field with a static string. Select Replace value and enter the string to use:

- Use the simplest possible regular expressions and grok matching patterns. Do not make excessive use of wildcards or quantifiers as it can slow the responsiveness of the Controller UI. Examples of expensive and greedy quantifiers can be found here: https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html
11. On the Agent Mapping tab, assign the source rule to specific Agent Scopes.
12. Click Save when you are finished. The source rule is saved in a disabled state.

Centralized Log Management UI Details

**General Configuration Tab**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Source Name</strong></td>
<td>Name of this source rule. This name must be unique. Appears in the list of defined source rules.</td>
</tr>
<tr>
<td><strong>Source Type</strong></td>
<td>Specifies the event type of log source file. This field is prepopulated when you start from an AppDynamics template or rule. If you are creating a new source rule from scratch, you can specify any value. This value is used to identify this source event and can be used in searching and filtering the collected log data. The value shows up in the field list for the log data.</td>
</tr>
<tr>
<td><strong>Source File</strong></td>
<td>The location and name of the log file to serve as a log source. The location must be on the same machine as the analytics-agent. You can use wild cards and you can specify whether to match files one level deep or all log files in the path directory structure. You can use wild cards and you can specify whether to match files one level deep or all log files in the path directory structure.</td>
</tr>
<tr>
<td><strong>Exclude Files</strong></td>
<td>Exclude, or blacklist, files from the defined source rule(s). Input the relative path of the file(s) to exclude, using wildcard names.</td>
</tr>
<tr>
<td><strong>TCP Port</strong></td>
<td>Specifies the port for the analytics-agent to collect log files from a network connection. This field is not present if the collection type is local file system. If no port number is provided, port 514 is used. Both the syslog utility and analytics-agent should have root access to send logs to port 514 (binding to ports less than 1024 requires root access).</td>
</tr>
<tr>
<td><strong>Enable path extraction</strong></td>
<td>This checkbox enables extraction of fields from the path name. Add a grok pattern in the text box. The syntax for a grok pattern is {%SYNTAX:SEMANTIC}. For example, to extract AdminServer from /opt/apps/oracle/middleware/user_projects/domains/ouaf_domain1/servers/AdminServer/logs/AdminServer.log, enter the following GROK pattern: /user_projects/domains/ouaf_domain1/servers/AdminServer/logs/AdminServer.log</td>
</tr>
<tr>
<td><strong>Start collecting from</strong></td>
<td>Indicate where to begin tailing (collecting the log records). Options are:</td>
</tr>
<tr>
<td></td>
<td>• From the beginning of the log file</td>
</tr>
<tr>
<td></td>
<td>• From the end of the log file</td>
</tr>
<tr>
<td></td>
<td>• A specific time range (in hours) - For example, if you set this to four hours, then when you enable the rule and the log data, only the last four hours of data is ingested. When tailing starts, the watermark state is maintained for is now stopped and restarted, it will start tailing from where it left off for any old file. For new files, it will only tail the current file.</td>
</tr>
<tr>
<td><strong>Override time zone</strong></td>
<td>Use this to override the time zone for eventTimestamp in log events. If your log events don't have a time zone assigned, you can override it, the host machine time zone is assigned to the log events. You cannot override the timezone in the pickup file. If you override the time zone, you must provide a timestamp format. Time zone formats should conform to Joda-Time, and the timestamp format should be created with the Joda-Time formatting system.</td>
</tr>
</tbody>
</table>
Override timestamp format

You can override the format if needed. Time zone formats should conform to Joda-Time Available Time Zones, and should be created with the Joda-Time formatting system.

Auto-Correct duplicate Timestamps

Enable this option to preserve the order of original log messages if the messages contain duplicate timestamps. This preserve the order.

Collect Gzip files

Enable this to also find gzip files in the specified path.

Field Extraction Tab

The following table describes the fields and actions on this tab. For a detailed procedure on using Auto or Manual Field Extraction, see Field Extraction for Source Rules.

<table>
<thead>
<tr>
<th>Section/Field or Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add Grok Pattern</td>
<td></td>
</tr>
<tr>
<td>Message Pattern</td>
<td>This is the grok pattern used to extract fields from the log message. A pattern may be prepopulated when you use an AppDynamics template or an existing source rule as your starting point. You can add or remove grok patterns as needed.</td>
</tr>
</tbody>
</table>
| Multiline Format        | For log files that include log records that span multiple lines (spanning multiple line breaks), use this field to indicate how the individual records in the log file should be identified. You have two options:  
  - **startsWith**: A simple prefix that matches the start of the multiline log record.  
  - **regex**: A regular expression that matches the multiline log record.  
The multiline format is not supported when you are collecting log data from a network connection. |
| Extract Key-Value Pairs |             |
| Field                   | Shows the field selected for key-value pair configuration. |
| Split                   | The delimiter used to separate the key from its value. In this example, "key=value": the split delimiter is the equal sign "=". Multiple comma-separated values can be added. |
| Separator               | The delimiter used to separate out two key-value pairs. In this example, "key1=value1;key2=value2": the separator is the semi-colon ";". Multiple comma-separated values can be added. |
| Trim                    | A list of characters to remove from the starting and/or the end of the key/value before storing them. In this example, "_ThreadID_": you can specify the underscore "_" as the trim character to result in "ThreadID". Multiple comma-separated values can be added. |
| Include                 | A list of key names to capture from the "source". You must provide keys in the include field. If the include field is left blank no key-value pairs are collected. |
| Actions                 |             |
| Upload Sample file      | Browse to a local file to upload a sample log file. |
### Preview
Use to refresh the Preview grid to see the results of the specified field extraction pattern.

### Auto Field Extraction

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Definer Sample</strong></td>
<td>Select a message from the preview grid that is representative of the fields that you want to extract from the log records. You can select only one definer sample per source rule.</td>
</tr>
<tr>
<td><strong>Refiner Samples</strong></td>
<td>Refines the regular expression to include values that were not included in the original definer sample.</td>
</tr>
<tr>
<td><strong>Counter Sample</strong></td>
<td>This specifies something to ignore. Refines the regular expression to exclude values that were included by the original definer or refiner sample.</td>
</tr>
</tbody>
</table>

### Manual Field Extraction

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Regular Expression</strong></td>
<td>Add a regular expression to define the field you want to extract.</td>
</tr>
<tr>
<td><strong>Field Type</strong></td>
<td>Specify the type for the field.</td>
</tr>
<tr>
<td><strong>Field Name</strong></td>
<td>Automatically generated within the regular expression pattern. This name appears in the Fields list in the Analytics Search UI.</td>
</tr>
</tbody>
</table>

### Actions

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Add Field</strong></td>
<td>Use to add additional regular expressions for extracting more fields. You cannot add more than 20 fields.</td>
</tr>
<tr>
<td><strong>Preview</strong></td>
<td>Use these buttons to filter the viewable results in the preview grid.</td>
</tr>
<tr>
<td>- All</td>
<td></td>
</tr>
<tr>
<td>- Matching</td>
<td></td>
</tr>
<tr>
<td>- Non-Matching</td>
<td></td>
</tr>
<tr>
<td><strong>Upload Sample File</strong></td>
<td>Upload a sample file from your local file system to use in the preview grid.</td>
</tr>
</tbody>
</table>

### Field Management Tab

<table>
<thead>
<tr>
<th>Field/Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Select Any Field to Customize</strong></td>
<td>Select the field to add customizations. You can add multiple customizations to a single field.</td>
</tr>
<tr>
<td><strong>Field Name</strong></td>
<td>This column lists the fields that have customizations.</td>
</tr>
<tr>
<td><strong>Customize</strong></td>
<td>This column shows the specific customizations. For static fields, the display name is shown. Static fields cannot be further customized.</td>
</tr>
<tr>
<td><strong>Mask Value</strong></td>
<td>This customization option masks values in the collected data. Specify the starting and ending position in the data and the character to use as the masking value.</td>
</tr>
<tr>
<td><strong>Replace Value</strong></td>
<td>This customization option replaces the entire value of the field with a static string.</td>
</tr>
<tr>
<td><strong>Rename</strong></td>
<td>This customization option enables you to rename a field to a more recognizable display name.</td>
</tr>
<tr>
<td><strong>Field Type</strong></td>
<td>Use to change the data type of the field. For example, from string to number. Available types are String, Boolean, and Number. Note that after a source rule has been saved with a field of a specified data type, you cannot later change the data type of that field. Once the fields are indexed in the analytics database trying to specify a new type will cause a validation error.</td>
</tr>
<tr>
<td><strong>Remove</strong></td>
<td>This customization turns off data collection for the field. It can be reversed at a later time.</td>
</tr>
<tr>
<td><strong>Add Static Field</strong></td>
<td>This action allows you to add a static field to all the log events collected from this source. This field can then be used to search and filter the log data. For example, use this to add Tier, Node, and Application Name to the log data.</td>
</tr>
</tbody>
</table>
Field Extraction for Source Rules

This topic provides a detailed procedure for how to use Auto and Manual Field Extraction to configure Log Analytics source rules. Field Extraction uses regular expressions to identify and format the fields for your Log Analytics data.

Auto Field Extraction enables you to upload a sample log file and select fields for extraction. The necessary regular expressions are automatically generated and highlighted in your sample messages. You can fine-tune the generated regular expressions using Refiner Sample and Counter Sample log messages.

Manul Field Extraction enables you to upload a sample log file and enter your own regular expressions to define fields and associate the fields with a data type.

**Definitions**

- **Source Rule**: a set of configuration settings for extracting analytics data from log files. You create source rules using the Centralized Log Management UI in the Controller. Source rules are stored in the Controller data store and periodically synced with the Analytics Agent. Also, see Configure Log Analytics Using Source Rules.
- **Sample File**: a representative log file uploaded to the Controller that provides a way to test and fine-tune your log file source rules.
- **Definer Sample**: a specific log message selected from the sample log file used to define the fields that you want to extract from the log messages.
- **Refiner Sample**: an additional log message used to revise the auto-generated regular expression created in the Definer Sample step. The Refiner sample helps capture fields that were missed in the initial step.
- **Counter Sample**: an additional log message used to eliminate false positives while extracting the fields as defined in the Definer or Refiner steps.

**Auto Field Extraction**

This section describes how to use Auto Field Extraction to extract fields from your log files.

To specify field extraction using a Definer Sample log message:

1. Using the Centralized Log Management UI, add a new source rule (or edit an existing source rule), and navigate to the Auto Field Extraction subtab within the Field Extraction tab. See the section "Create a Source Rule" in Configure Log Analytics Using Source Rules for detailed steps on accessing this UI.
2. If you do not see log messages in the grid, upload a sample file by clicking Upload Sample File.
   Note - if you uploaded a sample file in a previous step in the configuration process or you started your new source rule using an existing source rule, you will see log messages in the grid.
3. Hover over any row on the grid and click +Select as Definer Sample.
   Choose a sample log message that contains the fields you want to extract from the log file. When you need to capture fields from dissimilar log messages, you can create more than one definer.
4. After you select the Definer Sample message, use the following steps to extract fields from the message. After selecting the Definer Sample and specifying the fields to extract, the preview grid displays rows with matching and non-matching results.
   a. Select text in the message. If you are defining key-value pairs, it is only necessary to highlight the key field.
      A popup appears where you can specify properties of the extracted field.
   b. Enter the field name, select the data type, and click Extract.
      The preview grid is updated to show matching rows with the selected text highlighted with a color.
   c. (Optional) To update the properties or delete the field, click the field you just created in the definer and the properties popup appears.
   d. Repeat steps a, b and c to create more fields.
      As you add additional fields, the regular expression changes. You can view or copy (but not edit) the regular expression in the regular expression section.
5. (Optional) Specify Refiner Samples: As you review the non-matching rows, if you notice a value that was not extracted and should have been extracted, then you can add that row as a Refiner Sample. A refiner sample is used to match the unmatched rows from the Definer step.
   a. On the grid, click <strong>Select as Refiner Sample</strong>. You do not see this option unless you already selected the Definer Sample.
   b. Select the text.
   c. A popup appears. You can associate this portion of text with a field already created in the Definer step. When you extract, the regular expression changes and matching rows get updated in the Preview grid and will be shown as Matching. If necessary, you can add more than one Refiner Sample message.

6. (Optional) Specify Counter Samples: Counter samples eliminate false positives while extracting the fields. In the Preview grid, the highlighted values are shown along with an X button. Click the X next to a field to mark a value as a false positive (a Counter Sample).

The Counter Sample is shown below the Definer and Refiner Samples and the regular expression is updated to show the new set of matching and non-matching rows.

7. Click <strong>Save</strong> to save the fields to the source rule.
8. Continue to the other tabs of the Add Source Rule dialog to complete your source rule configuration.

**Manual Field Extraction**
You can specify your own regular expressions if needed using Manual Field Extraction.

1. Using the Centralized Log Management UI, add a new source rule (or edit an existing source rule), and navigate to the Manual Field Extraction subtab within the Field Extraction tab. See the section “Create a Source Rule” in Configure Log Analytics Using Source Rules for detailed steps on accessing this UI.

2. If you do not see log messages in the grid, upload a sample file by clicking **Upload Sample File**.

3. Click **Add Field**.

4. Click **Show Example** to review how to write regular expressions.

5. Type in your regular expression and associate it with a field type using the drop-down menu.

6. In the Preview grid, click **Refresh**.
   The preview grid is highlighted with the results of applying the regular expression to the sample log messages. You cannot add more than 20 fields.

7. (Optional) You can change the regular expression or the type and use **Refresh** to see the updated results.

8. Click Save to update your source rule.
Agent Scopes for Configuration Log Management

On this page:
- Static Agent Scopes
- Dynamic Agent Scopes
- Define Agent Scopes

Related pages:
- Configure Log Analytics Using Source Rules

Agent Scopes are used to determine which agents use the CLM source rules. You can group your deployed Analytics Agents into scopes, then assign source rules to the scopes. While creating an agent scope, you are given an option to either choose agents from a static list or to add agents dynamically by specifying matching rules.

Static Agent Scopes

The Available Agents list is populated with the Analytics Agents reporting to your controller. You can use this list to add currently available agents to a scope.

Dynamic Agent Scopes

You can specify match conditions where registered Analytics Agents that match the rule are automatically added to, or excluded from, the scope. Analytics Agents that register with your Controller in the future, with characteristics that match the scope criteria, will also be added to the scope. You can match on agent name, unique host ID, or IP address.

The available operators for the matching conditions include regular expressions (regex) and Classless Inter-Domain Routing (CIDR) notation. CIDR is a compact notation for an IP address and its associated routing prefix.

The match conditions are additive, so exclusions will take precedence over inclusions. An agent must satisfy all conditions to be included. For example, an agent named ABC would not be included in the scope if you specify something similar to the following conditions:
- include name equals ‘ABC’
- include name that ends with ‘D’

Define Agent Scopes

To add an Agent Scope

1. From the Controller top navigation bar, click Analytics.
2. From the left navigation panel, click Configuration > Log Analytics > Agent Scopes.
3. **Click +Add.**
   You see the Add Agent Scope pane where you can specify how you want to add agents: from a static list or dynamically by specifying match rules.

   ![Add Agent Scope](image)

   - Add agents to scope by
     - Choosing specific agents
     - Specifying rules to match agents

   ![Next button](image)

4. To add specific agents, select **Choosing specific agents** and move agents from the list of Available Agents to the list of Selected Agents to define the scope.
5. To add agents dynamically, select **Specifying rules to match agents** and specify your matching rules.
6. Name the Agent Scope and Save.
Collect Log Analytics Data from Syslog Messages

On this page:

- Supported Environment Details
- Send syslog Data Directly to the Analytics Agent
- Read syslog Messages From the Log File and Send to Analytics Agent
- Configure Log Analytics TCP Source Rule
- Configure Log Analytics TCP Job File

Related pages:

- Configure Log Analytics Using Job Files
- Configure Log Analytics Using Source Rules
- Analytics Log Data

The Analytics Agent can be configured to receive syslog messages using TCP transport and standard syslog format adhering to RFC 5424. The Analytics Agent can listen on a TCP port for syslog messages coming from a remote source or from the same host on which the Analytics Agent is present.

There are two steps to set this up:

- Configure your web server, such as Apache, to send logs to the Analytics Agent. There are two ways to direct the syslog data to the Analytics Agent:
  - Write the log messages to the usual log files and then read and forward to the agent from the log file. See Read syslog Messages From the Log File and Send to Analytics Agent
  - Send the log data directly to the Analytics Agent without writing to a file first. See Send syslog Data Directly to the Analytics Agent.
- Configure the Analytics Agent to receive and parse the logs. There are two ways to configure syslog message collection for Log Analytics:
  - Job files (used in 4.2 and prior), see Configure Log Analytics TCP Job File.
  - Source rules (new in 4.3) Recommended for your new log file sources. See Configure Log Analytics TCP Source Rule.

Supported Environment Details

- Linux only.
- Network protocol is TCP only
- A template for Apache Web Server syslog format is shipped with the Analytics Agent. You can bring any log data in through syslog over TCP as long as you provide the correct configuration to parse the log message.
- One job file per Analytics Agent can be enabled to receive syslog messages over TCP.

Both the syslog utility and analytics-agent should have root access if the port where the analytics-agent is listening for syslog messages is lower than 1024.

Send syslog Data Directly to the Analytics Agent

You can use the Linux Logger utility to direct your Apache access and error log messages to the syslog daemon directly. Using this method, the logs are not written to the usual access log and error log apache files or to the /var/log/message file. Instead, the log lines are sent directly to a syslog daemon which then forwards the data to the analytics-agent.

Configure the following:

1. Configure Apache to delegate logs to /usr/bin/logger.
   a. Locate and open httpd.conf, the Apache configuration file. This is typically located at /etc/httpd/conf/.
   b. Add a new CustomLog directive to send access logs and error logs to the syslog and comment out the line that sends to the access_logs.
1. Comment the following line to avoid logging to access_logs:

```
#CustomLog logs/access_log combined
```

2. Add a new CustomLog directive to send access logs and error logs to the syslog:

```
CustomLog "|/usr/bin/logger -t httpd -p local6.info" combined
```

This directive uses the logger utility to send messages with facility local6, tag 'httpd' and log format combined. The facility code specifies the type of program that is logging the message. Messages with different facilities may be handled differently. The tag 'httpd' in the httpd.conf directly relates to the program name in the rsyslog.conf (see next step) for filtering out which messages need to be sent. For example, there can be other programs writing to that particular port, but we only want to send the logs from the httpd program to analytics-agent.

2. Configure the rsyslog client.
   a. Edit rsyslog.conf, typically located at /etc/.
   b. Add the following lines above "RULES" or "var/log/messages" filter so Apache doesn't log to the /var/log/messages file:

   ```
   # log to analytics-agent
   if $syslogfacility-text == 'local6' and $programname == 'httpd'
   then @<analytics_agent_home>:514
   # Prevent logging httpd to /var/log/messages
   if $syslogfacility-text == 'local6' and $programname == 'httpd'
   then ~
   ```

   Make sure you replace 514 with the port where the analytics-agent is listening for syslog messages. This must match the port specified in the job file.

3. Restart apache and rsyslog, and look at /var/log/messages for any rsyslog errors.

Read syslog Messages From the Log File and Send to Analytics Agent

In this case, the Apache server writes logs to the usual access log and error log files, and the rsyslog daemon is configured to read from these log files and forward the log data to analytics agent. This method preserves the original log files. In this case, the access and the analytics logs are logged in the /var/log/messages file.

In this example, the rsyslog client is configured to read from a specific file and forward the message with facility local6 and severity level info over the specified port (default port is 514).

1. Locate and edit rsyslog.conf, typically located at /etc/.
2. In the begin forwarding section of the rsyslog.conf file, add the following lines:
2. Add these lines in the begin forwarding section:

```
# add these lines in the begin forwarding section
$ModLoad imfile
$InputFileName /etc/httpd/logs/access_log << your file
$InputFileTag apache-access
$InputFileStateFile stat-apache-access
$InputFileSeverity info
$InputFileFacility local6
$InputRunFileMonitor
local6.info @@localhost:514
```

- **InputFileName**: path to the log file you want to tail.
- **local6.info**: Use the Analytics Agent IP address if your analytics agent is not local to the controller. If needed, replace 514 with the port where the analytics-agent is listening for syslog messages. This must match the port specified in the job file or the source rule.

3. Restart rsyslog and review /var/log/messages for any rsyslog errors.

### Configure Log Analytics TCP Source Rule

Using the Centralized Log Management UI, you can configure a source rule to extract log analytics fields from syslog messages over TCP.

1. Access the Centralized Log Management UI from your Controller by clicking **Analytics > Configuration > Log Analytics**.
2. On the Source Rules tab, click **+ Add**.
3. In the Add Source Rule panel, select the **Create from** source template and select **From Network Connection** as the collection type. For example, select the default Apache syslog template `apache-httpserver-access-syslog`:

![Add Source Rule](image)

Several log format templates are shipped with the Analytics Agent. You can create a new source rule for any log format over syslog TCP as long as you configure it correctly.

4. Click **Next** to see the Add Source Configuration wizard.
5. Specify the collection details, such as the name of the source rule, source type and enter the TCP Port where the Analytics Agent is listening.
6. When you specify **From Network Connection** as the collection type, the grok pattern for the syslog header (which is appended to the log messages) is automatically added at the beginning of the grok Message Pattern:
7. Confirm that the value for Multiline Format is **None**.
8. Configure field extraction and field management as for any other source rule. See Configure Log Analytics Using Source Rules for more details.

**Configure Log Analytics TCP Job File**

When selecting and configuring the port where the analytics agent will listen for the syslog data, make sure it does not conflict with anything else active in the network. If no port number is provided, port 514 is used. Both the syslog utility and analytics-agent should have root access to send logs to port 514 (binding to ports less than 1024 requires root access).

To allow the analytics-agent to listen at a port, specify the log file `source` property and associated parameters for `type=syslog`. For example, add the following to the appropriate job file:

```yaml
source:
  type: syslog
  port: 514
  protocol: tcp
  numThreads: 1
```

A job file for apache commons is included in the analytics distribution at `/<analytics-agent-home>/conf/job/sample-apache-httpserver-access-syslog.job`.

Your job file should look similar to the following:
version: 2
enabled: true

source:
  type: syslog
  port: 514
  protocol: tcp
numThreads: 5

fields:
  sourceType: apache-httpserver-access-syslog
  nodeName: Node1
  tierName: Tier1
  appName: App1

grok:
  patterns:
    - "%(SYSLOG5424PRI)%{SYSLOGBASE2} %{COMBINEDAPACHELOG}"

eventTimestamp:
  pattern: "dd/MMM/yyyy:HH:mm:ss Z"
Configure Log Analytics Using Job Files

This topic describes how to configure log sources using job files. If you are configuring new log sources, we recommend you use the Centralized Log Management UI to define source rules.

The general steps to configure log analytics using job files are:

1. Describe the Log Source in a Job File
2. Map the Log File Data to Analytics Fields
3. Verify Analytics Agent Properties

Describe the Log Source in a Job File

Each log source is represented by a job file. A job file is a configuration file that specifies the following:

- Location of the source log file
- Pattern for capturing records from the log file
- Pattern for structuring the data from the captured log records
- Other options for capturing records from the log source

To define a source, you create a job file (or modify one of the samples) in the Analytics Agent configuration directory. The Analytics Agent includes sample job files for Glassfish, OSX log, and others. The Analytics Agent can also collect and parse GZIP files - (log files ending in .gz).

The job files are located in the following directory:

```bash
<Analytics_Agent_Home>/conf/job/
```

The agent reads the job files in the directory dynamically, so you can add job files in the directory without restarting the agent.

To configure a job file, use the following configurable settings in the file:

- **enabled**: Determines whether this log source is active. To capture analytics data from this log source, set the value to true.
- **source**: Specifies the source of the logs.
  - **type**: Specifies the type of log source. There are two types, file, and syslog. Additional parameters depend on the value of **type**.
  - **file**: The location and name of the log file to serve as a log source. The location must be on the same machine as the analytics-agent. File has the following parameters:
    - **path**: Path to the directory where the log files reside. On Windows, the path should be provided as if on Unix environments such as:
      - Example: demo/logs
      - Example: C:/app/logs
    - **nameGlob**: A string to use to match on the log file name. You can use wild cards and you can specify whether to match files one level deep or all log files in the path directory structure. If the wild card starts the value of nameGlob, you must enclose the value in quotes. The matching patterns that are supported can be found here: [http://java.boot.by/ocjp7-upgrade/ch06s05.html](http://java.boot.by/ocjp7-upgrade/ch06s05.html) under "glob".
      - Example for multi-level matching:
        ```
        path: /var/log
        nameGlob: "*/.log"
        ```
        This matches both /var/log/apache2/logs/error.log and /var/log/cassandra/system.log
Example of one level matching:
path: /var/log
nameglob: "/\.*\.log"
This searches for .log files one level deep in the /var/log directory (matches on /var/log/cassandra/system.log but not on /var/log/apache2/logs/error.log).

- **startAtEnd**: If set to true allows tailing the file from the end.
- **syslog**: For details on using syslog, see [Collect Log Analytics Data from Syslog Messages](#).

- **multiline**: For log files that include log records that span multiple lines (spanning multiple line breaks) configure the multiline property and indicate how the individual records in the log file should be identified. A typical example of a multiline log record is one that includes a Java exception. You can use one of two options with the multiline property to identify the lines for a multiline log record:
  - **startsWith**: A simple prefix that matches the start of the multiline log record.
    Example 1: To capture the following multiline log as one record:
    ```
    [#|2015-09-24T06:33:31.574-0700|INFO|glassfish3.1.2|com.appdynamics.METRICS.WRITE|_ThreadID=206;_ThreadName=Thread-2;|NODE PURGER Completed in 14 ms|#]
    ```
    You could use this:
    ```
    multiline:
    startsWith: "[#]"
    ```
    Example 2: To capture the following multiline log as one record:
May 5, 2016 6:07:02 AM
com.appdynamics.test.payments.plugin.testsource.testPaymentClient deposit(pluginContext,
financialTransaction, retry)
INFO: Source deposit requested amount:245.43000
May 5, 2016 6:07:02 AM
com.appdynamics.test.payments.plugin.testsource.testPaymentClient deposit(pluginContext,
financialTransaction, retry)
INFO: Source deposit currency:USD
May 5, 2016 6:07:02 AM
com.appdynamics.test.payments.plugin.testsource.testPaymentClient deposit(pluginContext,
financialTransaction, retry)
INFO: Source deposit payment method:Master Card
May 5, 2016 6:07:03 AM
com.appdynamics.test.payments.plugin.testsource.testPaymentClient deposit(pluginContext,
financialTransaction, retry)
INFO: Source deposit decision:ACCEPT
May 5, 2016 6:07:03 AM
com.appdynamics.test.payments.plugin.testsource.testPaymentClient deposit(pluginContext,
financialTransaction, retry)
INFO: Source deposit reason code:200
May 5, 2016 6:07:03 AM
com.appdynamics.test.payments.plugin.testsource.testPaymentClient deposit(pluginContext,
financialTransaction, retry)
INFO: Source deposit orderId:7654
May 5, 2016 6:07:03 AM
com.appdynamics.test.payments.plugin.testsource.testPaymentClient deposit(pluginContext,
financialTransaction, retry)
INFO: Source deposit piId:1234B
May 5, 2016 6:07:03 AM
com.appdynamics.test.payments.plugin.testsource.testPaymentClient deposit(pluginContext,
financialTransaction, retry)
INFO: Source deposit referenceNumber:4620346703956702001
May 5, 2016 6:07:03 AM
com.appdynamics.test.payments.plugin.testsource.testPaymentClient deposit(pluginContext,
financialTransaction, retry)
INFO: Source deposit extData
requestToken:alf/7wSR9PBh3zSMQ+++IYTR1k3Ys2DOxOm5jWLRyFJaSggCTyFJaSgjSB1/2v0wyTWRV0ekb0X
You could use this:

```
multiline:
    startsWith: "INFO: Source deposit requested amount"
```

- **regex**: A regular expression that matches the multiline log record.
  
  **Example**: To capture this multiline log as one record:

```
=======================================
merchantID=appD, sendToProduction=true,
targetAPIVersion=1.00, keyFilename=(null),
serverURL=(null), namespaceURI=(null), enableLog=true,
logDirectory=logs, logFilename=(null),
logMaximumSize=10, useHttpClient=false, timeout=130
2016-06-01 16:28:21.037 WebContainer : 8 REQUEST >
merchantID=appD
davService_run=true
clientLibraryVersion=2.0.1
clientEnvironment=OS/400/V7R1M0/LINUX
application_id=ff22
exportService_addressWeight=medium
merchantReferenceCode=appD
clientLibrary=Java Basic
billTo_customer=xxxxxxxxx
billTo_finance=true
billTo_postalCode=94105
billTo_country=US
```

You can use the following regex configuration to identify the start of each "record":

```
multiline:
    regex: "^\d{4}-\d{2}-\d{2}\:\d{2}:\d{2}\:\d{3} WebContainer : \d+ TRANSTART\u003E
    $"
```

The regex describes that the line should match 4 digits(2016) followed by followed by 2 digits (01), followed by a space, followed by 2 digit(16) "": 2 digit (28) "": 2 digits(21) "": 3 digits(035) space followed by term 'WebContainer' space digit followed by term 'TRANSTART' folowed by '>'.
Whenever the Log tailer sees the matching pattern at the start of the line, it starts a new match and passes the previously collected data as one log record. Each matching pattern identifies when to start accumulating the log lines in a buffer until the matching pattern is found again at the beginning of another record.

**Note:** If the particular format of a multiline log file does not permit reliable continuation line matching by regular expression, you may choose to use a single line format. For most types of logs, this would result in the capture of the majority of log records.

- **fields:** The fields are used to specify the context of the log data in the Controller UI, by application name, tier name, and so on. Specify the fields as free form, key-value pairs.
- **grok:** The grok parameter specifies the patterns by which the data in the unstructured log record is mapped to structured analytics fields. It associates a named grok expression (as defined in a .grok file that is bundled inside `lib/analytics-shared-pipeline-core.jar`) to a field in the data as structured by the agent. For example:

```
grok:
  patterns:
    - \\
      \[%{LOGLEVEL:logLevel}%{SPACE}\\
      \[%{DATA:threadName}\\]  \\\%{JAVACLASS:class}\\
      %{GREEDYDATA}"
    - "pattern 2"
...
```

In this case, the grok-pattern name LOGLEVEL is matched to an analytics data field named logLevel. The regular expression that is specified by the name LOGLEVEL is defined in the file `grok-patterns.grok` in the `grok` directory. For more about Grok expressions, see **Specifying Grok Expressions**.

Previous versions of Log Analytics used a single “pattern” rather than a pattern list. This mode is still supported for backward compatibility.

- **keyValue:** The keyValue parameter specifies how to parse the logs to identify key-value pairs with a user-defined delimiter. This enables you to configure the parsing for a message of the type “Key1 = Value1 Key2 = Value 2”. For more details, see **Specifying Key Value Pairs**.
- **transform:** This parameter specifies how to change the type or alias name of any field extracted from the logs by grok or keyValue parameters.
- **eventTimestamp:** This setting defines the pattern for the timestamp associated with captured data.

### Map the Log File Data to Analytics Fields

To specify how data in the unstructured log records should be mapped to structured analytics fields for log analytics, you provide the configuration in the job file. You can map unstructured log data in the following ways:

- **grok patterns**
- **key value pairs**
- **transforms**

**Specifying Grok Expressions**

Grok is a way to define and use complex, nested regular expressions in an easy to read and use format. Regular expressions defining discrete elements in a log file are mapped to grok-pattern names, which can also be used to create more complex patterns.

Grok-pattern names for many of the common types of data found in logs are provided for you with the analytics agent. A list of basic grok-pattern names and their underlying structures are bundled inside `lib/analytics-shared-pipeline-core.jar`. You can list the grok files with a command such as the following:
unzip -l ./lib/analytics-shared-pipeline-core.jar|grep "\.grok"

To can view the definition of a grok file with a command such as the following:

unzip -p ./lib/analytics-shared-pipeline-core.jar grok/grok-patterns.grok

The grok directory also contains samples of more complex definitions customized for various common log types - java.grok, mongodb.grok, and so on. Additional grok patterns can be found here: https://grokdebug.herokuapp.com/patterns#.

Once the grok-pattern names are created, they are then associated in the jobs file with field identifiers that become the analytics keys. The basic building block is %{grok-pattern name:identifier}, where grok-pattern name is the grok pattern that knows about the type of data in the log you want to fetch (based on a regex definition) and identifier is your identifier for the kind of data, which becomes the analytics key. So %{IP:client} would select an IP address in the log record and map it to the key client.

Custom grok patterns

Complex grok patterns can be created using nested basic patterns. For example, from the mongodb.grok file:

MONGO_LOG %{SYSLOGTIMESTAMP:timestamp} \[%{WORD:component}\] %{GREEDYDATA}

It is also possible to create entirely new patterns using regular expressions. For example, the following line from java.grok defines a grok pattern named JAVACLASS.

JAVACLASS \(?:[a-zA-Z$\_][a-zA-Z\$\_0-9]*\.[a-zA-Z\$\_][a-zA-Z\$\_0-9]*/

Because JAVACLASS is defined in a .grok file in the grok directory can be used as if it were a basic grok pattern. In a jobs file, you can use the JAVACLASS pattern match as follows:

grok:
  pattern: ".... \[%{JAVACLASS:class}\]"

In this case, the field name as it appears in the Application Analytics UI would be "class". For a full example, see the following files:

- Job file: <Analytics_Agent_Home>/conf/job/sample-analytics-log.job
- Grok file: java.grok bundled inside lib/analytics-shared-pipeline-core.jar

Special Considerations for Backslashes

The job file is in YAML format, which treats the backslash as an escape character. Therefore, to include a literal backslash in the String pattern you need to escape the backslash with a second backslash. You can avoid the need to escape backslashes in the .job file grok pattern, by enclosing the grok pattern in single quotes instead of double quotes such as the following:
Numeric Fields

In Release 4.1.3, the grok definition syntax was enhanced to support three basic data types. When defining a pattern in the .grok file you can specify the data type as number, boolean, or string. If a Grok alias uses that grok definition in a .job file then the extracted field is stored as a number or boolean. Strings are the default. If the number or boolean conversion fails, then a log message appears in the agent's log file. No validations are performed as it is not possible to reverse engineer a regex reliably. These are pure runtime extractions and conversions.

Upgrade pre-4.1.3 Job Files

For 4.1.2 (or older) .job files in use that have fields that are unspecified or specified as NUMBER and now switch to the "type aware" files, the data inside Events Service will break. This is due to the type mapping. To avoid this, you need to modify the grok alias in your job files.

Examples:

Was:
```grok
patterns:
- '\[%{DATESTAMP:TIME}\] %{SPACE}%{NOTSPACE:appId}%{SPACE}%{NOTSPACE:appName}%{SPACE}%{NOTSPACE:Severity}%{SPACE}%{NOTSPACE:messageId}: %{GREEDYDATA}'
```

Update job to:
```grok
patterns:
- '\[%{DATESTAMP:TIME}\] %{SPACE}%{NOTSPACE:appId}%{SPACE}%{NOTSPACE:appName}%{SPACE}%{NOTSPACE:Severity}%{SPACE}%{NOTSPACE:messageId}: %GREEDYDATA'
```
To Upgrade (migrate) pre-4.1.3 job files:

1. Stop analytics-agent.
2. Change .job files that use the enhanced grok patterns:

   BOOL:boolean
   INT:number
   BASE10NUM:number
   NUMBER:number
   POSINT:number
   NONNEGINT:number

Change the grok alias so as not to conflict with the older aliases:

   grok:
     patterns:
       (Old) -
       "%{DATE:quoteDate},%{NUMBER:open},%{NUMBER:high},%{NUMBER:low},%{NUMBER:close},%{NUMBER:volume},%{NUMBER:adjClose}"

       (New aliases) -
       "%{DATE:quoteDate},%{NUMBER:openNum},%{NUMBER:highNum},%{NUMBER:lowNum},%{NUMBER:closeNum},%{NUMBER:volumeNum},%{NUMBER:adjCloseNum}"


**Specifying Key-Value Pairs**

This section of the mapping configuration captures key-value pairs from fields specified by the `source` parameter. The values listed under `source` should refer to fields that were defined and captured by a grok pattern. For example, if you have a grok parameter that defines the following pattern `"%{DATA:startsWith}"` then you can list the field "startsWith" under the source parameter to extract key-value pairs from the entire log message. The result can be different than expected if the message contains more information than just key-value pairs.

The Key Value mapping contains the following fields:

- **source**: A list of strings on which the keyValue filter should be applied. It is an optional field. If it is not provided the key-value pairs are parsed from the original log "message" string.
- **split**: The delimiter defined by the user to separate out the key from the value. In this example, `key=value`, the split delimiter between the key and the value is the equal sign `=`
- **separator**: The delimiter defined by the user to separate out two key-value pairs. In this example, `key1=value1;key2=value2`, the separator is the semi-colon `;`
- **include**: A list of key names that the user wants to capture from the "source". If nothing is provided in "include" we capture all the key-value pairs.
- **trim**: A list of characters the user wants to remove from the starting and/or the end of the key/value before storing them.

The sample-glassfish-log.job job file includes key-value pairs configuration. This file is found here: `<analytics_agent_home>/conf/job/`
Key-Value Pairs Example

For a log file with the following entries:

```
[#I2015-09-24T06:33:31:574-0700|INFO|glassfish3.1.2|com.appdynamics,METRICS.WRITE|ThreadID=200;_ThreadName=Thread-2; NODE PURGER Complete in 14 ms|#]
```

And the following grok pattern:

```
grok:
  patterns:
    - "\\[\\[\\#\\]|%{DATA}| %{LOGLEVEL:logLevel}| %{DATA:serverVersion}| %{JAVACLASS:class}| %{DATA:keyValuePairs}| %{GREEDYDATA}"```

The key-value parameter to extract the ThreadID and the ThreadName should look similar to the following:

```
keyValue:
  source:
    - "keyValuePairs"
  split: "=
  separator: ";"
  include:
    - "ThreadID"
    - "ThreadName"
  trim:
    - " _"```

Specifying Transform Parameters

This section of the mapping configuration enables you to change the type or alias name of any field previously extracted from the logs by your grok or key value configuration. The transform is applied after all fields have been captured from the log message. You can specify a list of field names, where you want to cast the value to a specific type or rename the field with an "alias".

The Transform mapping contains the following fields:

- **field**: Specifies the name of the field to transform and can not be empty or null. If field is defined, either type or alias must be specified. If neither is specified, an error is written to the analytics-agent.log file.
- **alias**: The new name for the field.
- **type**: The value type for the field. Valid values are:
  - NUMBER
  - BOOLEAN
  - STRING - default is STRING

Verify Analytics Agent Properties

In addition to configuring the log source in the job file as described above, you should verify the values in the analytics-agent.pro
properties file found in the conf directory. Confirm the following property values:

- `http.event.endpoint` should be the location of the Events Service.
  - For SaaS controllers the URL is one of the following:
    - https://analytics.api.appdynamics.com:443 (North America)
    - https://fra-ana-api.saas.appdynamics.com:443 (Europe)
    - https://syd-ana-api.saas.appdynamics.com:443 (APAC)
  - For on-premises installations use whatever host and port you have configured. In clustered environments, this is often a load balancer.
- The `http.event.accountName` and `http.event.accessKey` settings should be set to the name and the key of the account in the Controller UI with which the logs should be associated. By default, they are set to the built-in account for a single tenancy Controller.
- The `pipeline.poll.dir` setting specifies where the log configuration files are located. This would not normally be changed unless you want to keep your files in a different location.
- `ad.controller.url` should match your AppDynamics controller URL and port.

Troubleshoot Logs

If log capture is working correctly, logs should start appearing in the Log tab in the Analytics UI. It can take some time for logs to start accumulating. Note the following troubleshooting points:

- If nothing appears in the log view, try searching over the past 24 hours.
- Timezone discrepancies between the logs and the local machine can cause log entries to be incorrectly excluded based on the selected timeframe in the Controller UI. To remediate, try setting the log files and system time to UTC or logging the timezone with the log message to verify.
- An inherent delay in indexing may result in the “last minute” view in the UI consistently yielding no logs. Increase the time range if you encounter this issue.

Troubleshoot Patterns

To help you troubleshoot the data extraction patterns in your job file, you can use the two debug REST endpoints in the Analytics Agent:

- `http://<Analytics_Agent_host>:<Analytics_Agent_http_port>/debug/grok`: For testing grok patterns
- `http://<Analytics_Agent_host>:<Analytics_Agent_http_port>/debug/timestamp`: For testing timestamp patterns

In the following examples, the Analytics Agent host is assumed to be localhost and the Analytics Agent port is assumed to be 9090. To configure the port on your Agent, use the property `ad.dw.http.port` in `<Analytics_Agent_Home>/conf/analytics-agent.properties`.

The Grok Endpoint

Click to expand...

The Grok tool works in two modes: extraction from a single line log and extraction from a multi-line log. To get a description of usage options:

```
curl -X GET http://localhost:9090/debug/grok
```

Single Line

In this mode, you pass in (as a POST request) a sample line from your log and the grok pattern you are testing, and you receive back the data you passed in organized as key/value pairs, where the keys are your identifiers.
curl -X POST http://localhost:9090/debug/grok --data-urlencode "logLine=LOG_LINE" --data-urlencode "pattern=_PATTERN"

For example, the input:

[o.e.j.server.handler.ContextHandler] Started
i.d.j.MutableServletContextHandler@2b3b527{/,,null,AVALAIBLE]"
--data-urlencode "pattern=\[%{LOGLEVEL:logLevel}%{SPACE}\]
\[%{DATA:threadName}\] \[%{JAVACLASS:class}\] \%{GREEDYDATA}\"

would produce this output:

```json
{
    threadName => main
    logLevel => INFO
    class => o.e.j.server.handler.ContextHandler
}
```

The input:

curl -X POST http://localhost:9090/debug/grok --data-urlencode "logLine=2010-05-05,500.98,515.72,500.47,509.76,4566900,509.76"

would produce this output:
Multi-line

The multi-line version uses a file stored on the local filesystem as the source input.

```bash
curl -X POST http://localhost:9090/debug/grok --data-urlencode "logLine=`cat FILE_NAME`" --data-urlencode "pattern=PATTERN"
```

where FILE_NAME is the full path filename of the file that contains the multi-line log.

**The Timestamp Endpoint**

Click to expand...

The timestamp tool extracts the timestamp from a log line in Unix epoch time.

To get a description of usage options:

```bash
curl -X GET http://localhost:9090/debug/timestamp
```

In this mode, you pass in (as a POST request) a sample line from your log and the timestamp pattern you are testing, and you receive back the timestamp contained within the log line.

```bash
curl -X POST http://localhost:9090/debug/timestamp --data-urlencode "logLine=LOG_LINE" --data-urlencode "pattern=PATTERN"
```

For example, the input:

```json
{
  open => 500.98
  adjClose => 509.76
  volume => 4566900
  quoteDate => 10-05-05
  high => 515.72
  low => 500.47
  close => 509.76
}
```
 curl -X POST http://localhost:9090/debug/timestamp
  [main] [io.dropwizard.server.ServerFactory] Starting DemoMain"
  --data-urlencode "pattern=yyyy-MM-dd'T'HH:mm:ss,SSSZ"

would produce this output Unix epoch time:

{  
  eventTimestamp => 1409844161237 
}

The input:

 curl -X POST http://localhost:9090/debug/timestamp
  --data-urlencode "logLine=Nov 17, 2014 8:21:51 AM
  com.foo.blitz.processor.core.hbase.coprocessor.endpoint.TimeRollup
  ProcessEndpoint$HBaseDataFetcher callFoo1" --data-urlencode  
  "pattern=MMM d, yyyy h:mm:ss aa"

would produce this output Unix epoch time:

{  
  eventTimestamp => 1416212511000 
}
Sample Log Analytics Job Files

A number of sample job files are shipped with the Analytics Agent. These files can be found at `<analytics-agent-home>/conf/job`.

The sample job files include the following:

- Analytics logs: sample-analytics-log.job
- Analytics logs with requestGUID config example: sample-analytics-log-with-request-guid.job
- Apache access logs: sample-apache-httpserver-access-log.job
- Apache access logs with syslog config example: sample-apache-httpserver-access-sysloglog.job
- Apache error logs: sample-apache-httpserver-error-log.job
- Cassandra logs: sample-cassandra-log.job
- CouchDB logs: sample-couchdb-log.job
- Glassfish logs: sample-glassfish-log.job
- IIS logs: sample-iis-log.job
- Java Agent logs: sample-java-agent-log.job
- Jetty error logs: sample-jetty-error-log.job
- Jetty request logs: sample-jetty-request-log.job
- Log4J: sample-log4j.job
- MongoDB logs: sample-mongodb-log.job
- MySQL error logs: sample-mysql-error-log.job
- Nginx access logs: sample-nginx-access-log.job
- Nginx error logs: sample-nginx-error-log.job
- OS X system logs: sample-osx-system-log.job
- Postgres logs: sample-postgres-log.job
- Redis logs: sample-redis-log.job
- Stock quotes: sample-stock-quotes-csv.job
- WebLogic logs: sample-weblogic-log.job
Migrate Log Analytics Job Files to Source Rules

The Centralized Log Management UI enables you to configure your log data sources using source rules. Manually created job files are no longer needed.

We recommend the following steps to convert your log source configuration from job files (job files were used for configuration before version 4.3) to source rules.

1. Confirm that you have upgraded your Controller, Events Service and Analytics Agent to version 4.4. See the corresponding upgrade topics for details:
   - Upgrade the Controller Using the Enterprise Console
   - Upgrade the Events Service Using the Enterprise Console
   - Upgrade Analytics Agent
2. Use the Centralized Log Management UI to create source rules for your existing job files by importing the configuration from your old job file to a new source rule.

Be sure to use either ‘Parsing time range’ or ‘End of log file’. If you do not use this, you will double collect all the log files that have already been tailed by the job file. Read Timing Notes for more information to help you decide on your collection timing settings.

When you save the source rules, they are in the disabled state by default.

3. Map the source rules to an Agent Scope.
4. Enable the new source rules through the UI.
5. Disable the job files that are actively collecting log analytics data. You do this by manually editing the job file and changing the enabled property to false. See Configure Log Analytics Using Job Files.
6. After you have migrated all your job file configurations to source rules, you can completely disable the use of job files by clicking ‘Disable Field Extraction With Job Files’. This action can not be reversed and you will not be able to use job files after performing this action. Only trigger this action once you have completely moved to source rules created through the Centralized Log Management UI.
Timing Notes

By default, the controller communicates new configuration information to the Analytics Agent every five minutes. So it could be up to five minutes before the agent starts tailing your log file using the new source rule configuration. On the other hand, if you have both a job file and a source rule enabled for the same log data, the data is collected twice. To avoid this situation, you can use configuration settings to specify when to start collecting the log data:

- at the end of the file (UI field = **Start collecting from End of log file**)
- during a specific time range. (UI field = **Parsing time range**)

See the section describing the configuration settings on the "**General Configuration Tab**" in Configure Log Analytics Using Source Rules.

Be sure to disable the job file once you have enabled the source rule to collect the Log Analytics data. You can verify that the new source rule is collecting data correctly by waiting until the new log records appear in the Analytics Search UI data grid. One way to distinguish between the log data collected by a job file or a source rule is to use a different source type in the source rule.
When you investigate the causes of slowdowns or outages in your business applications, the problem does not always originate in the application code. Any additional information from application logs or machine logs can be helpful for your DevOps teams in investigating the problem. One way to see the supporting data that is impacting the business transaction is to analyze logs for that transaction.

By configuring business transactions for GUID Injection, you can correlate logs to specific business transaction requests. This can be helpful when you see slow transactions and the call graph does not give you enough information to get to the root cause. You can use this feature to get the full context related to a failed or slow transaction.

Correlating specific instances of your business transactions to the related logs works by injecting the same requestGUID of the business transaction into the associated logs through our Java Agent. This helps you to quickly find the relevant logs from multiple tiers and nodes for a specific business transaction.

Visibility of this correlation requires a license for both Transaction Analytics and Log Analytics.

### Configuring GUID Injection

This feature is supported for two Java logging frameworks:

- Apache log4j versions 1.2.12 to 1.2.17
- logback versions 1.0.0 to 1.1.3. Any version before 1.0.0 is not supported.

To enable transaction to log correlation, use the following steps. You must select the business transactions and specify the logging format.

1. Select business transactions for log correlation.
   
   This is how you specify to the Java Agent which business transactions you are interested in.
   
   a. In the Controller UI, click Analytics > Configuration. Then select Log Analytics > Logging Transaction Correlation.
   
   b. Select the application from the dropdown list:

   ![Configuration](image)

   c. Scroll down to the section **Configure Transactions for GUID Injection**:
d. Add transactions from the right-hand list to the left-hand list and **Save**.

e. Proceed to the next step. You must also configure the logging patterns before you can see correlated logs.

2. Define how the injected information appears in the logs. You need to know the appender name and pattern for your application logging framework so you can properly configure this feature. In simple terms, a **pattern** is responsible for formatting a logging request and an **appender** takes care of output destination. You can configure this through the Controller UI as described here. You can also add the appender directly to your code. See **Add Appender in Your Source Code**.

   a. On the Analytics Configuration window, scroll to the section **Configure Patterns for Logging format**.

   b. Select the application, tier, or node where you want to collect correlated logs.

   **Configure Patterns for Logging format**

   Select Application, Tier, Node

   ![Select Application, Tier, Node](image)

   ![ECommerce-Sales-Demo](image)

   ![Add Log Appender](image)

   ![Apply to all Nodes](image)

   ![Copy to](image)

   ![Save](image)

   c. Enter the Appender name and choose the log framework.
2. c. d. Enter the pattern and the request GUID string. You can add the request GUID anyplace in the pattern. The request GUID must match the following exactly:

```
[%X{AD.requestGUID}]
```

For example, the following screenshot shows the standard log4j pattern plus the request GUID:

```
Add Log Appender
```
```
Appender Name
```
```
ECommerce-appender
```
```
Type
```
```
Log4j
```
```
Pattern
```
```
[%d] [%-5p] [%t] [%c] [%X{AD.requestGUID}] %m%n
```

3. Restart the affected application to enable the logger to pick up the new logging configuration.
Add Appender in Your Source Code

If you have access to the source code for your application, you can also add the appender to the log4j.properties file directly. Here is an example of what that might look like with the request GUID in **BOLD**:

```properties
log4j.appender.order-file-appender=org.apache.log4j.FileAppender
log4j.appender.order-file-appender.File=logs/telecom-order.log
log4j.appender.order-file-appender.layout=org.apache.log4j.PatternLayout
log4j.appender.order-file-appender.layout.ConversionPattern=[%d] [%-5p] [%t] [%c] [%X{AD.requestGUID}] %m%n
log4j.logger.com.appdynamics.order=DEBUG, order-file-appender
log4j.additivity.com.appdynamics.order=false
```

Viewing Correlated Logs and Transaction Data

Once you have configured GUID injection, you can search the log files from a number of points in the UI. Correlation only works when there are logs with the associated GUID in the given time range for an application where this feature is configured. Therefore, the search button, **Search Logs By Request GUID**, appears when there are logs for the snapshot request GUID in the snapshot time range.

APM to Analytics: From a business transaction snapshot

1. Locate a slow transaction or other transaction that you want to troubleshoot.

   ![Selection of a slow transaction](image)

Select and double-click a slow transaction.

2. Click **More Details > Search Logs** or use **Actions > Search Logs By Request GUID**.
From the log details in this example, you can see that the reason this transaction was slow is that the order processing queue was full and it took many retries before the order could complete.
Transaction Analytics to Log Analytics

You can select relevant transactions from any transaction analytics search and see details.

1. Select the transaction, then double-click or click Details in the Action toolbar.

2. On the Overview tab, you have two ways to search the logs by request GUID.
2. When you search the logs, you see the correlated logs in the Log Analytics window.

3. When you search the logs, you see the correlated logs in the Log Analytics window.

From Log Analytics to Transaction Analytics

You can select a log from any log analytics search and see the Event Details.

1. Select the log, then double-click or click Details in the Action toolbar.
   You see the Event Details window.
2. Use the Search Transactions button to find the relevant business transaction in transaction analytics data.
Using Application Analytics Data

This page provides an overview of how to use the Application Analytics data. You have several strategies available for locating and using analytics data of interest. Once you have identified data that is useful for your business, you can use visualization or other features to represent and monitor that data.

The Analytics UI enables you to:

- Configure data sources to collect data for analytics
- Create and visualize searches and add them to custom dashboards
- Create API Keys for users of the Analytics Events API
- Create and view metrics created from analytics searches
- Define and monitor Business Journeys
- Define Experience Level reports
- Export data to CSV files

### Strategies for Locating Data of Interest

- Use a saved search by selecting one from the list
  - Use the actions in the action bar to manage named searches (saving, duplicating, and so on)
- Create searches for specific use cases. For details, see Search Analytics Data.
- Focus on a specific time range by dragging your mouse across the event stream or by using the time range dropdown
- Select various fields from the data, targeting the kind of data of interest to you, then scan the event list:
  - Double-click on any specific event to display detailed information
- Examine Top 10 Values. Click on a field to see the top 10 values of that field in your filtered dataset. The results are presented as a count and percentage of all data within the specified time range for that field. This can help you get immediate insights from your data without having any predefined rules or previous knowledge of the data.
  - You can add a value to the search criteria bar by hovering over the value and clicking the plus icon that appears to the right.
- Relevant Fields. This feature helps you find fields with a high relevance score. A high relevance score indicates these fields are significantly more common in your filtered results than in the entire data set and may be useful to investigate. See Investigate Using Relevant Fields for a suggested workflow.

### Visualize Analytics Data

Once you have defined your data, use the Visualization tab to explore the specific aspects of the data that interest you and to drill down into the relationships you need to understand.

Each visualization type is a widget, and widgets can be added and removed as desired. For details, see Visualize Analytics Data.

### Upgrade Visualizations from 4.1

If you are upgrading from 4.1, you can convert your Saved Searches using Actions > Import From 4.1 in the action bar.
Don't click Import From 4.1 more than once. There is a known issue that the Saved Searches from 4.1 can be imported multiple times if you click Import from 4.1 multiple times.

Importing from 4.1 converts the following widgets:

- Column
- Pie
- Table
- Treemap
- Time Series Line
- Time Series Area
- Time Series Scatter Plot
- Time Series Column
- Numeric widgets built with the Custom Widget Builder
Search Analytics Data

You can search Analytics data in two ways:

- **Basic mode**, where you can add search criteria to filter your data and return a subset of available events. In Basic mode, you see the list of events that match your search criteria, along with a count of those events. This mode is referred to in the UI as Drag and Drop Search.
- **Advanced Mode**, where you can use the AppDynamics Query Language (ADQL). ADQL is a SQL-like query language that provides additional operators and functions to enable more complicated searches. This mode is referred to in the UI as Query Language Search.

For each mode, you see the analytics data sources that are licensed and enabled for your application. The applications, data sources, and other fields that you see depend on how the administrator has set the permissions for your role. When creating new roles, remember that granting permissions to view transaction analytics data does not automatically grant permissions to see all application data associated with a specific transaction analytics record. You need to grant at least read-only permissions to the application to enable the user to see associated transaction snapshot data such as flow maps.

### Search Results Limits

Search results in the UI can return up to 1000 records. By default, each page displays 50 results at a time. In Advanced mode, you can explicitly set a LIMIT to see more than 50 results on a page. Setting a limit disables pagination and shows up to 1000 records on a single page. The total number of records for the query shows at the bottom right. The UI caps the number of search results at 1000 regardless of mode or the value of an explicit LIMIT. To return more than 1000 records, you must use the Analytics Events Query API. See LIMIT Clause for more details.

### Exporting Analytics Data

Application Analytics allows you to export data to a file in CSV format for external use.

Before you export a set of data, consider the following guidelines:

- The maximum number of records exported for a non-aggregation query is 65000. Use the LIMIT clause to restrict the records to a number below the upper bound. The LIMIT clause does not return any additional records that satisfy the export selection criteria.
- The number of records exported for an aggregation query by default is 10. If you want more than 10 records, use the LIMIT clause. You can export up to 1000 records by using the LIMIT clause.
- The CSV file format is RFC4180.
- Response from all valid ADQL queries can be exported. The method of writing ADQL queries remains the same.
- Only the fields that you specify in the SELECT clause are exported to the CSV file. ADQL query supersedes the UI selection. The fields selected to be displayed in the Data tab has no impact on the exported data even if the query includes SELECT *.
- Fields having nested representation are flattened and appear as separate fields in the CSV file. Additionally, the corresponding unflattened nested fields also appear in the exported records. Flattening fields refers to separating nested fields into individual fields.
- The supported datetime format is ISO, yyyy-MM-dd'T'HH:mm:ssZ, without the millisecond value. The timezone is local to the browser from which export is requested. A timestamp field in a nested field is converted to the local timezone only in the flattened fields, not in the unflattened nested field. The format in the UI, that is MM/DD/YYYY HH:MM:SS AM/PM, is different from the format that you see in the exported data.
- The time required to export data varies with the size of the record you are trying to export. Exporting tends to slow down for larger set of records because executing the query on and retrieving a large data set takes longer to complete.
You can export metrics from the Metric Browser and search results from the Searches screen. Use the Export option under Actions on the toolbar to do so. While you are on the Searches screen, use the Data mode to export your search result.

Access the Analytics Search UI

1. Navigate to the Analytics interface in the Controller UI by clicking Analytics in the top navigation bar.

2. Click Searches to create new searches and see a list of previously saved searches.

   The saved searches list displays any pre-existing, saved searches for which you have view permissions. The default sort order for the list is alphabetical by saved search name.
3. To view your saved searches, click **View Options** and select **Filter by Current User**.

4. Click **+Add** to create a new search or select an existing search to activate the actions on the action toolbar.

5. When you click **+Add**, you see options to select the search mode.

   - Drag and Drop Search, see [Create Basic Analytics Searches](#)
   - Query Language Search: see [Create Advanced Analytics Searches](#)
On the Search details page, you see two tabs for displaying the results for the Search.

- All event types share the same basic layout on the Data tab of the Search details screen.
- Use the Data tab to search and filter the data that you want to review and analyze.
- Use the Visualization tab to create graphic representations based on that data.
- Search actions are available for each search. The actions vary according to the type of search and the tab you use. Generally, the following actions are available:
  - Rename
  - Export (to CVS - only from the Data tab)
  - Create Metric
  - Add to Dashboard (only from the Visualization tab)

### Basic and Advanced Search Comparison

The following table provides a comparison of the functionality available in each search mode.

<table>
<thead>
<tr>
<th>Functionality</th>
<th>Basic</th>
<th>Advanced</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drag and drop fields to the criteria search bar</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Type-ahead (auto-complete)</td>
<td>N/A</td>
<td>yes</td>
</tr>
<tr>
<td>Visualization canvas widget support</td>
<td>multiple</td>
<td>one widget per search query</td>
</tr>
<tr>
<td>Grouping of events with the GROUP BY keyword</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Wild cards with comma-separated text (Basic mode) or IN operator (Advanced mode)</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Percentile Histogram widget</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>
Create Advanced Analytics Searches

On this page:
- Constructing ADQL Queries
- Using ADQL Auto-Complete
- Field Name Mapping

Related pages:
- Create Basic Analytics Searches
- Using Application Analytics Data
- ADQL Reference

This topic describes using the AppDynamics Query Language (ADQL) to search analytics data using Advanced mode in the Analytics Search window. If you have never used AppDynamics Analytics, you might want to start with basic mode before using Advanced mode for your searches.

Access Advanced mode by clicking Add+ on the Analytics Search page. Select Query Language Search, as indicated in the following screen capture.

**Constructing ADQL Queries**

Each ADQL event query begins with a SELECT statement that uses a FROM clause. Use the FROM clause to specify the data source you want to query. You can query one data source in each query.

Use the SELECT statement to further qualify the query using field_expressions and math_expressions. The WHERE clause adds conditions. For more details on specific ADQL syntax, keywords, functions, and operators, see the ADQL Reference.

When you click Search for queries other than select all fields (SELECT *), the results display on the visualization tab where the table widget accurately renders your query.

**Using ADQL Auto-Complete**

As you construct your query and type a space after a keyword, auto-complete offers suggestions for the next keyword or operator in the query. The display of auto-complete suggestions is based on the stage of your query and the event type you selected. Auto-complete offers suggestions for data values, such as event types and fields, when you type an equals sign “=” or other comparison operator after the event type name.

Note: The query box in the data tab is used only for adding WHERE clause differentiation. Aggregation functions are only available in the visualization tab under the table widget.

In advanced mode, when you type SELECT, the fields from the last specified event type display in the auto-complete list. Fields specific
to a new event type display after you complete the **FROM** clause.

Remember to type a space in front of the next keyword or you may see a validation error when executing the search or the search will be inexecutable.

**Field Name Mapping**

The field names in the auto-complete dropdown list correspond to UI labels and are obvious in most cases with minor differences in capitalization. When the field names have prepended database-related information, such as `segments.errorList.errorType` for Error Type or `segments.httpData` for HTTP data fields, they refer to data for individual segments of the transaction as it progresses from tier to tier. The segments fields do not represent data that refers to the overall business transaction instance. See **ADQL Data** for a list of fields, UI labels, and internal database names for each data source.
Create Basic Analytics Searches

On this page:
- UI Elements
- Basic Search
- Search Options
- Free Text Search in Log Analytics

Related pages:
- Create Advanced Analytics Searches

This topic describes UI elements available in the Analytics search screens and focuses on how to create searches in Basic mode. This mode is also called Drag and Drop Search in the UI and enables you to do multi-widget visualization of your search results. In this mode, you add search criteria based on the fields collected for the event type. Field search enables you to search for all values or a specific value of the field.

UI Elements

The following UI elements are available to help construct your searches:

- **Query bar:** consists of the event type selector and the Add Criteria button.

  ![New Search](image)

  Click the down arrow to see the drop-down list for the available event types. Click Add Criteria to see the available fields for the selected event type.

- **Search box:** The scope of the search depends on the context. For example, you can search for a field in the field list to locate a specific field:
For some fields, such as business transaction, you can enter a term to narrow the list to values containing that term:
Hover over a specific numeric field to view basic statistics associated with the field:

- **Field value selector:** After you have selected a field, you may see a drop-down that shows a list of values for that field. For example:
Text filter: Enables you to do an exact match on the term entered. Supports wild cards. For other options, see Search Options.

Combination field selector and search: This field value selector combines a list of values with the ability to search for terms in the values list. It also has an "Advanced" option that displays the text filter instead.
Clicking Advanced Filter and the down arrow again displays a text filter:

Basic Search

To create a search using a field:

1. From the Analytics Search list, click + Add.
2. To use basic mode and multi-widget visualizations, select Drag and Drop Search.
3. On the Data tab, select the data source you want to search.
4. To add filters to your search, click **Add Criteria**.
The field selector drop-down appears, so you can select the field you want to use.

5. Select a field. The field name appears in the search criteria line.
6. Click the drop-down next to the field name to open the field value selector. Select a value for the field.

Fields such as application, business transaction, and node display a list of the values available. Some fields show a search box where you can enter criteria to filter the values of this field.

### Search Options

Free text search and text filter behavior are described in the following table.

<table>
<thead>
<tr>
<th>Type of entry</th>
<th>Analyzed</th>
<th>Non-analyzed</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>term</code></td>
<td>Find entries with this term. This search is case insensitive. For example, using the term &quot;great&quot; finds &quot;great&quot; and &quot;AppDynamics is great&quot;.</td>
<td>Looks for an exact match and is case sensitive. For example, using the term &quot;order&quot; finds &quot;order&quot; but not &quot;/PlaceOrder&quot;.</td>
</tr>
<tr>
<td><code>term1 term2</code></td>
<td>Returns results containing both <code>term1</code> and <code>term2</code> irrespective of the order the terms appear.</td>
<td>Terms are implicitly AND'd. Looks for an exact match and find results containing &quot;term1 term2&quot;.</td>
</tr>
<tr>
<td><code>term1, term2</code></td>
<td>This works the same as the space-separated list of terms.</td>
<td>Terms are implicitly OR'd. Looks for an exact match for either <code>term1</code> or <code>term2</code>.</td>
</tr>
<tr>
<td><code>term*</code></td>
<td>Finds all fields where the value is this string plus one additional character <code>&lt;&gt;</code> or multiple <code>&lt;*&gt;</code>. Because all non-alphanumeric fields are delimiters, the escape character has no effect.</td>
<td>Finds all fields where the value is this string plus one additional character <code>&lt;&gt;</code> or multiple <code>&lt;*&gt;</code>. To search for a field containing a wild card character, escape with a &quot;:&quot;.&quot;</td>
</tr>
<tr>
<td>NOT term</td>
<td>Use the <strong>NOT</strong> comparison operator to find everything except the term. <strong>NOT</strong> can be used with any of the entries in this table.</td>
<td>Use the <strong>NOT</strong> comparison operator to find everything except the term. <strong>NOT</strong> can be used with any of the entries in this table.</td>
</tr>
</tbody>
</table>

**Free Text Search in Log Analytics**

For log analytics, a free text search is available for searching the message field. This enables you to search for any keyword or string anywhere in the message field.
Investigate Using Relevant Fields

You can use Relevant Fields to find data fields that show a high Relevance Score. A high relevance score indicates these fields are significantly more common in your filtered dataset than in the entire dataset and may be particularly useful to analyze. Relevant field investigation may provide valuable insights and enable you to conduct better root cause analysis.

Relevant Fields investigation can be used with all event types.

As an example, you can use the following suggested workflow to investigate poor user experience.

1. From the Analytics Search page, select the Transactions event type and the Data tab.

2. Click Add Criteria and select an application to investigate.

3. Click User Experience or another field in the Fields list to see the Top Values. If you see lots of error and very slow transactions, you realize it is not easy to investigate hundreds of error and very slow transactions.

4. Add Error and Very Slow values to your search criteria to narrow the total number of transactions to a smaller dataset. You can add the values quickly by hovering over the field in the Top Values list until the + sign appears at the right. Click the + to add to the search criteria. Now the focus is only on error/very slow transactions.
5. Click the **Relevant Fields** label and now you can see where to focus your investigation. In this example, you might start with the **Checkout** business transaction as the relevance score is 87. You can also see that **Platinum** and **Silver** customers have
the two highest scores. The relevance score is an indication of where the highest proportion of error and very slow transactions are occurring.

6. Click a field in the Relevant Field list to see more details. You can continue to add relevant fields to your search criteria to keep narrowing the dataset. This process should enable you to find issues of significance for investigation.
Visualize Analytics Data

On this page:
- Access the Analytics Widget Builders
- Build a Widget
- Conversion Analysis Using Funnel Widget
- Configure Custom Widgets
- Custom Widget Builder Data Type and Field Restrictions

Related pages:
- Create Basic Analytics Searches

After you have created a search for the Analytics data that you want to visualize, you are ready to configure a widget to display the results of the search graphically.

Access the Analytics Widget Builders

You can access the Analytics visualization widgets from Analytics Search UI or directly from Dashboards & Reports.

To access the Analytics widget builder from Analytics

1. At the top of the Controller UI, select Analytics > Searches to view the saved searches list.
2. Double-click the search that you want to visualize or add a new search.

To access the Analytics widget builder from a Custom Dashboard

1. From the top navigation bar, click Dashboards & Reports to see the custom dashboard list.
2. Either double-click the dashboard where you want to add the Analytics widget or click Create Dashboard to create a new dashboard.
3. In the custom dashboard click Edit, then Add Widget.
4. Click Analytics, Saved Searches, or Experience Levels.
5. Add Analytics widgets as needed to your dashboard.

Build a Widget

To use the Analytics widget builder:

1. Click the Visualization tab.
2. In the Visualization canvas, you can create widgets with one of the following modes:
   - In Drag and Drop Search (Basic) mode you can create multiple widgets on the canvas.
   - In Query Language Search (Advanced) mode you can generate one widget from your ADQL query at a time. In addition, when you use the Visualization tab in this mode, the order of functions and fields in the query matters. For all widgets (except table), the aggregate functions, such as count(*), avg(), sum(), percentile and so on, should always be preceded by a field. If the fields and functions in the query are not in this order, some visualizations do not work.
   - For an example of creating a conversion analysis using the funnel widget, refer to Conversion Analysis.
3. Click Add Widget, then select Custom Widget Builder or Funnel Widget Builder depending on the type of widget that you want to create.
   - In Basic mode, you can drag fields to the canvas and build custom widgets, which use X and Y axes for configuration. After dragging a field to the axes, select a function from the drop-down next to the field name. In this example, the percentile function is applied to HTTP Time (ms).
3. The funnel widget is configured differently from the other custom widgets, which use Y and X axes as the basis of the configuration. A funnel widget is used to display data in progressively staged proportions in a color-coded funnel. You can add a title to the widget by over-writing **Widget Title** on the canvas.

4. After adding a widget and fields, you can right-click menu widget values to see other options. Select "Apply a global filter for `<field_value>`" to add the value to the existing search criteria. The supported boolean operators are **true, false, null**, and **is not null**. Use **is not null** to check both boolean and non-boolean fields, such as loan amount or IsReached.

5. Click **Save & Close** to save the widget configuration.

**Conversion Analysis Using Funnel Widget**

Funnel analysis involves using a series of events that lead towards a defined goal, such as from user activity in a mobile app to a sale on an eCommerce website or from advertisement to purchase in online advertising. With the funnel widget, you can visualize use cases such as:

- Retail application checkout drop-off rates from the point of adding items to the shopping cart to the actual purchase
- Business Journey drop-off rates from the point of applying for a loan to approving the loan
- Home mortgage loan approval drop-off rates from submitted application to approved applications
- Impact of performance issues by total sales, customer segment, location and products purchased can be gauged by using a funnel along with other health rule widgets as shown in this custom dashboard:

![Funnel Chart](image)

Funnels return the number of unique actors that successfully (or unsuccessfully) make it through a series of steps. **Actors** could mean users, devices, or any other identifiers that are meaningful to you. A funnel analysis reveals where a given flow loses the most actors. This helps identify areas for improvement, as well as the overall health of your business.

The first step in the funnel is the anchor step, and always represents 100% of the results. The filter criteria for the first step creates a set
of unique values, and every subsequent step results in a subset of the values from the previous step and renders the corresponding drop-offs. The second and later steps describe the number of events that match the criteria for each step towards the goal. This number is typically less than 100%, but could be 100% if every event matches the subsequent steps.

**Build a Funnel Widget**

1. From the Analytics Search page using Drag and Drop (Basic) Mode, select the Visualization tab.
2. Click **Add Widget > Funnel Widget Builder**.

   ![Funnel Widget Builder](image)

3. In the Funnel Definition pane, click the down arrow to select the unique field representing the items that you are counting. In this example, following the steps of a loan approval process, LoanId is used. The fields depend on your application. There is a limit of 50,000 unique values for the first step. If you encounter this limit, you can try decreasing the global time range and look at fewer values at a time.

   ![Funnel Definition Pane](image)

4. Label the first step and continue to add steps as needed for your application process.
For this example, the first step represents the number of submitted loan applications. The search criteria include the AD-Capital application and the `/portal/SubmitApplication` business transaction.

Alternatively, you can set `SubmitApps.milestoneReached` to `true`. 
5. Continue to add and label your steps. This step represents the number of loan applications that reached the stage "Documents Verified".

Alternatively, use the `milestoneReached` field and set it to `true`.
6. After you have added all the necessary steps, click **Add & Close Widget Builder** to save the completed funnel widget. This example contains two additional steps representing loans that reached the credit check and approval stages.

If you want to cancel editing and revert to the previous version, click **Close Widget Builder**.

If the funnel steps need a rearrangement you can simply reorder them by simply dragging and dropping each step into appropriate positions.

7. Once you finish building your funnel, remember to name and save your search result.
8. To add the finished funnel widget to a custom dashboard, click **Actions > Add to Dashboard**.

Using Funnel to Troubleshoot

The finished funnel widget for the above example shows the drop-offs (**Abandoners**) in unique loan IDs from submission to approval. When creating the funnel, you can also select **Show Health** in the properties pane to display a health overlay that correlates application health data with the business conversion data. This helps to visualize performance for each step of the funnel widget.
Using the health overlay is supported for events types that contain a user experience value:

- Transaction Events – userExperience
- Browser Records – pageexperience
- Mobile Snapshots (Network Requests) – networkrequestexperience
- Web Session Records (Browser Sessions) – experience
- Mobile Session – networkrequest.networkrequestexperience

For enhanced troubleshooting, you can drill down to event data from any of the funnel steps by double-clicking the numbers on conversions and abandoners, or by using health overlay and clicking on each health segment.

Configure Custom Widgets

To configure custom widgets for your analytics searches:

1. From the Analytics Search page using Drag and Drop Search (Basic) Mode, select the Visualization tab.
2. Click Add Widget > Custom Widget Builder.

3. To add a field to a widget, drag the field from the Fields panel on the left to the Y or X-axis on the right. The available fields are based on the fields returned by the search.

4. From the Chart Types panel on the right, click the widget type to use for the display. The widget is previewed in the main panel. You can experiment previewing which widget type is best for visualizing a particular data set.
4. Usage for the X and Y axes depends on the chart type. The UI prevents you from adding more fields than the widget supports. Different widgets have different limits for the maximum number of fields permitted on each axis.

- For the time series widgets (area, line, scatter, histogram), you can use the Resolution field that appears on the X-axis to customize the frequency displayed if you do not want the widget to automatically select a default resolution based on the current time range. This time resolution field is not counted toward the field limit. The granularity of resolutions that can be chosen depends on the Global time range.

### Custom Widget Builder Data Type and Field Restrictions

<table>
<thead>
<tr>
<th>Widget Type</th>
<th>Data Type</th>
<th>Number of fields on Y-axis</th>
<th>Number of fields on X-axis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column (simple)</td>
<td>any</td>
<td>zero</td>
<td>up to two</td>
</tr>
<tr>
<td></td>
<td></td>
<td>one</td>
<td>up to two</td>
</tr>
<tr>
<td></td>
<td></td>
<td>two or three</td>
<td>zero or one</td>
</tr>
<tr>
<td>Pie</td>
<td>any</td>
<td>one</td>
<td>up to two</td>
</tr>
<tr>
<td>Table</td>
<td>any</td>
<td>one</td>
<td>up to ten</td>
</tr>
<tr>
<td>Tree Map</td>
<td>any</td>
<td>one</td>
<td>up to two</td>
</tr>
<tr>
<td>Time Series (Area, Line, Points, Column)</td>
<td>any</td>
<td>zero</td>
<td>one</td>
</tr>
<tr>
<td></td>
<td></td>
<td>one</td>
<td>one</td>
</tr>
<tr>
<td></td>
<td></td>
<td>two or three</td>
<td>zero</td>
</tr>
<tr>
<td>Percentile Histogram</td>
<td>numeric</td>
<td>zero</td>
<td>one</td>
</tr>
<tr>
<td>(Not available in Advanced Mode)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Numeric (Univariate)</td>
<td>any</td>
<td>one</td>
<td>zero</td>
</tr>
</tbody>
</table>
Visualize JavaScript Errors

You can use Analytics to query your Browser Request event data and create a JavaScript exception and error summary dashboard.

The following procedure assumes that you have configured and enabled the reporting of the JavaScript errors for Browser RUM and that JavaScript errors exist in the Events Service.

Using ADQL, you can search for the errors and then export a chart visualizing the results to a custom dashboard.

1. From the Analytics tab in the Controller UI, click Add to access the Analytics Search page.
2. Select Advanced mode and enter the following query:
   ```sql
   SELECT scripterrordata.message, count(*) FROM browser_records WHERE scripterrordata.message IS NOT NULL.
   ```
   The results are shown on the visualization tab similar to the following:

   ![Visualization Chart](image)

3. Select the chart type you prefer to visualize the results.
4. From the Actions menu, click Add to Dashboard and either create a new dashboard or add it to your existing dashboard. Now you can customize the dashboard as needed with labels or other widgets and data.

You can also leverage additional fields such as scripterrordata.origin or pagename to group the errors together. The additional fields are the following:

- scripterrordata.linenumber
- scripterrordata.origin
- scripterrordata.timestamp
<table>
<thead>
<tr>
<th>scripterrordata.lineno</th>
<th>scripterrordata.message</th>
<th>scripterrordata.origin</th>
<th>count(*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>TypeError: md.parseint is not a function</td>
<td><a href="http://127.0.0.1/eum-jp-error.js">http://127.0.0.1/eum-jp-error.js</a></td>
<td>250</td>
</tr>
<tr>
<td>6</td>
<td>TypeError: cars is undefined</td>
<td><a href="http://127.0.0.1/eum-jp-error.js">http://127.0.0.1/eum-jp-error.js</a></td>
<td>244</td>
</tr>
<tr>
<td>6</td>
<td>Uncaught TypeError: Cannot read property 4 of undefined</td>
<td><a href="http://127.0.0.1/eum-jp-error.js">http://127.0.0.1/eum-jp-error.js</a></td>
<td>2</td>
</tr>
<tr>
<td>30</td>
<td>malformed URI sequence</td>
<td><a href="http://127.0.0.1/eum-error.html">http://127.0.0.1/eum-error.html</a></td>
<td>128</td>
</tr>
<tr>
<td>30</td>
<td>URIError: malformed URI sequence</td>
<td><a href="http://127.0.0.1/eum-error.html">http://127.0.0.1/eum-error.html</a></td>
<td>42</td>
</tr>
<tr>
<td>30</td>
<td>Uncaught URIError: URI malformed</td>
<td><a href="http://127.0.0.1/eum-error.html">http://127.0.0.1/eum-error.html</a></td>
<td>3</td>
</tr>
<tr>
<td>19</td>
<td>cars is undefined</td>
<td><a href="http://127.0.0.1/eum-error.html">http://127.0.0.1/eum-error.html</a></td>
<td>114</td>
</tr>
<tr>
<td>19</td>
<td>TypeError: cars is undefined</td>
<td><a href="http://127.0.0.1/eum-error.html">http://127.0.0.1/eum-error.html</a></td>
<td>48</td>
</tr>
<tr>
<td>19</td>
<td>Uncaught TypeError: Cannot read property 4 of undefined</td>
<td><a href="http://127.0.0.1/eum-error.html">http://127.0.0.1/eum-error.html</a></td>
<td>2</td>
</tr>
<tr>
<td>48</td>
<td>ReferenceError: eum is not defined</td>
<td><a href="http://127.0.0.1/eum-error.html">http://127.0.0.1/eum-error.html</a></td>
<td>151</td>
</tr>
<tr>
<td>48</td>
<td>Uncaught ReferenceError: eum is not defined</td>
<td><a href="http://127.0.0.1/eum-error.html">http://127.0.0.1/eum-error.html</a></td>
<td>6</td>
</tr>
<tr>
<td>29</td>
<td>ReferenceError: missingFunction is not defined</td>
<td><a href="http://127.0.0.1/eum-jp-error.js">http://127.0.0.1/eum-jp-error.js</a></td>
<td>148</td>
</tr>
</tbody>
</table>
Create Analytics Metrics From Scheduled Queries

If you want to execute an analytics search repeatedly to monitor its value, you can create a metric from the search. The search will execute once per minute and report the results as a metric. You can create alerts on the metric in the usual way using Health Rules to trigger Policies and Actions. The analytics metric list is searchable.

You can create metrics for all Analytics event types - Browser, Mobile, Transaction, Log, and Custom.

Although the metrics calculated by scheduled queries may have decimal point precision, we only save and report whole numbers.

You can create a metric from an analytics search for the functions shown in the following table. For a discussion of metric rollup types see Metric Data Resolution over Time.

<table>
<thead>
<tr>
<th>Function</th>
<th>Metric Rollup Type</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>count(*)</td>
<td>sum</td>
<td>SELECT count(activeMacUsers) FROM dummyTransactions</td>
</tr>
<tr>
<td>distinctcount(field_name)</td>
<td>average</td>
<td>SELECT distinctcount(activeMacUsers) FROM dummyTransactions</td>
</tr>
<tr>
<td>sum(numeric_field_name)</td>
<td>sum</td>
<td>SELECT sum(responseTime) where userExperience = &quot;NORMAL&quot; FROM transactions</td>
</tr>
<tr>
<td>avg(numeric_field_name)</td>
<td>average</td>
<td>SELECT avg(responseTime) FROM transactions</td>
</tr>
<tr>
<td>min(numeric_field_name)</td>
<td>average</td>
<td>SELECT min(responseTime) FROM transactions</td>
</tr>
<tr>
<td>max(numeric_field_name)</td>
<td>average</td>
<td>SELECT max(responseTime) FROM transactions</td>
</tr>
<tr>
<td>(exp1)/(exp2)</td>
<td>average</td>
<td>SELECT (count(activeMacUsers)) / (avg(activeWindowsUsers)) FROM dummyTransactions</td>
</tr>
</tbody>
</table>

Each side of the expression must be enclosed in parenthesis.

Permissions

To create metrics from analytics searches, you need the following permissions:

- Manage Metrics permission
1. View access to the default Analytics Application

For more information, see:

- Analytics and Data Security
- Business iQ Analytics Permissions

Creating an Analytics Metric

1. Working in the Analytics > Search editor, set the search criteria to select the appropriate data.
2. Click Actions > Create Metric in the search action toolbar.

   
   ```sql
   SELECT count(*) FROM logs WHERE eventTimestamp > 1479414007000
   ```

3. In the pop-up window, give your metric a name and a description.
   The name determines how the metric appears in the Metrics screen and the Metric Browser. Keep in mind:
   - Metrics are truncated to whole values. So values less than one are truncated to zero. To work around this, you can multiply a numeric_field_name by a factor of 10, 100, 1000 and so on, depending on the level of decimal accuracy you need. We recommend that you add this factor into the metric display name so other users can understand what the metric represents, for example, Display Name = Metric (Factor 1000). In cases where the value is a percentage, and you convert by multiplying by 100, you probably don’t need to put the factor in the metric name because it would be implicit (Metric %).
   - Math operations are only supported inside the aggregation function. For example, \( \text{count} (\text{numeric_field_name} \times 10) \) from transactions.

Metric Timestamps

Timestamp metrics always aggregate events from the current minute because metrics are created for events published in the last minute.

If you have specified a value for the eventTimestamp field, the timestamp value in the new metric uses an aggregation of events from the pickupTimestamp field.

For example, you set an eventTimestamp for an event on a previous day. You then create a metric for the event today. The metric does not use your explicit eventTimestamp because it is no longer from the current minute. The metric aggregates events using pickupTimestamp. This ensures that the metric timestamp is based on the current minute because pickupTimestamp cannot be explicitly changed.

Learn more about timestamps in Analytics Events API.

Monitoring Analytics Metrics

For performance reasons, the scheduled queries used in analytics metrics can be disabled as follows:

- Queries are disabled if calling the Events Service results in any errors for ten consecutive times.
- Queries are not disabled if the Events Service is not reachable from the controller, such as during maintenance and upgrade windows.
- When the Controller is restarted, the queries which were previously disabled by the system due to consecutive failures are re-enabled.
- User-disabled queries remain disabled even after the system is restarted.
To monitor the metric:

1. Click **Metrics** in the left navigation bar. The Metrics list appears.

2. Search, edit, enable, disable, or delete metrics from here. You can edit only the description of the selected metric.

   To enable or disable metrics in a batch, select the corresponding rows in the grid and click the **Enable** or **Disable** button in the toolbar. Use Shift+ click or Command + click rows to select multiple rows.

3. To see the metric in the Metric Browser, click the Metric Browser. For more details, see the section "Metric Data Point Details" in Metric Browser.
Many industries have complex workflows and user journeys that span multiple transactions and event types, such as logs and End User Monitoring (EUM) data. These workflows typically take a long time to manifest and cannot be measured using transactions alone. Business Journeys are a way to monitor and correlate the data flow across multiple event sources and track the total end-to-end time for defined business workflows.

Typical multi-step workflows from different industries include:

- Payment transfers, credit card approval, and loan approval in financial services industries
- Cellphone activation and data recharge (pre-paid) in the Telco sector
- Insurance application through policy approval and insurance claims approval for insurance companies

**About Business Journeys**

An AppDynamics Business Journey is a composite event type based on defined Business Journey workflows. A Business Journey can include events from multiple analytics event types, such as logs, business transactions, custom events, and EUM data.

The following image depicts an example Business Journey definition, named loan approvals.

You create a Business Journey, as shown above, by defining the following:

- **Milestones** and the events comprising the milestones.
  Milestones are the steps in your business workflow.
Fields to be captured from each milestone event.  
A field indicates a category of information pertaining to the event.

The primary key field that uniquely ties the events together.  
A primary key correlates the milestones for your Business Journey. The primary key value must be present in each milestone event.

Additional fields that allow you to segment on different dimensions of the business workflow, such as loan types or payment amounts.

Health thresholds for monitoring the Business Journey performance.

Application Analytics starts collecting the Business Journey events after you have defined and enabled a Business Journey. Analytics does not go back in time to collect Business Journey composite events. To define specific Business Journeys, you may need to configure the collection of additional fields from your source analytics events. See Data Prerequisites for Defining Business Journeys for more information. Once the Business Journey events are being collected, you can view them from the Analytics Search UI in the same manner as other Analytics events. Because Business Journey events are, by their nature, potentially long-running processes, useful data might take some time to appear in the Analytics events list. Business Journey events are reported even when only one milestone is completed and the event is updated over time as subsequent events complete.

If you are already familiar with the fundamental concepts in Business Journeys clearly, proceed to either Configure Business Journeys or View Business Journeys as desired.

Business Journeys Milestones

A milestone is an event marking a significant stage in a business workflow. For example, in a loan application, the first milestone might be a user submitting a loan application. The second milestone might be document verification, followed by credit approval, insurance underwriting, and finally, loan approval. In each Business Journey milestone, you can add filters to specify the events that constitute a milestone and add fields to capture additional information about the workflow. You can use these events and fields to run ADQL queries and narrow down your searches.

As you create milestones, a flow map detailing the business workflow is automatically displayed in the Business Journey page. For more information, see Enable Flow Map.

When creating a Business Journey, you must use the same data type for the Primary Keys of all Milestones in that Business Journey.

Business Journeys Health Thresholds

A threshold is a boundary of acceptable or normal performance. Business Journey authoring provides default health thresholds against which it compares the performance of the business workflow. The health thresholds are based on the total time to complete all the milestones in your Business Journey. The threshold values are calculated by determining the standard deviation from the simple moving average over an interval of time. The default time interval is two hours. This means that if the average end-to-end time for the last two hours is N milliseconds, and if the Business Journey takes time equal to the standard deviation over N milliseconds (ms), it violates threshold.
Consider a simple moving average that is 1500 ms with a standard deviation of 100 ms. If you set the threshold to three, it means that the threshold is three times the standard deviation. In other words, a transaction that takes more than 1500 + (3*100) or 1800 ms violates the health threshold.

Before configuring the health threshold for a Business Journey definition, consider the following:

- The minimum duration over which standard deviation is calculated is 30 minutes.
- Total time for a Business Journey to complete is not an addition of the separate business transaction times. It is calculated by subtracting the timestamp for the first event from the timestamp for the final event.
- A user experience (normal, slow, very slow, stall) is assigned for the Business Journey event when a value for `totalTime` is present and is greater than zero. Total time for the event could end up less than zero if the milestones are defined in an incorrect order. This can also happen even after the order is correctly defined, but the milestones are not in time order such that an early milestone has a future date compared to a later milestone.
- The total time for the Business Journey is calculated when both the first and the last milestones events have been reached. Therefore, the user experience can not be determined and displayed until total time for the Business Journey is available.
- When a milestone encounters an error, for example, a transaction event where the requestExperience is Error, the user experience for the entire Business Journey event is marked as an Error. In this case, the total time will not be available.
- Typically when a milestone has an error, the entire Business Journey workflow does not complete. However, if for some reason, all the milestones complete even when milestone errors exist, the total time for the Business Journey is available. However, in this case, we will retain overall user experience as an error and not change to Normal, Slow, and so on.
- After a Business Journey is initially enabled, Business Journey events are generated. The standard deviation is calculated every five minutes, so the first few Business Journey events will not have a user experience assigned to them.

Filters and Fields

Filters allow you to extract event data for a select milestone based on selected criteria. For example, loan applications submitted in California. Filters define the scope of a milestone by limiting associated events. Fields extract additional relevant information from each milestone.

The mandatory filters are populated with the last 15 days of data. If no analytics data has been collected during the last 15 days, the drop-down menus for the mandatory fields will be empty. Besides the mandatory filters for each event type, Analytics allows you to optionally select additional filters by using the Filter By option. All the fields that are part of the filters and the additional filters that you optionally choose are automatically collected. You can use them to run ADQL queries later on.
A field indicates a category of information pertaining to the event. The Extract Fields section collects additional relevant information from each milestone. You can use data to query desired events and visualize matrices. The fields are categorized into mandatory and optional fields. The only mandatory field you must enter is the primary key. It is the unique identifier that ties milestones together that constitute a Business Journey.

In addition to the primary key, you can optionally add additional fields. These fields provide a context for your Business Journey definition. For example, loan amount, loan type, and customer name, customer id, and so on provide further details about a loan application. You can use these field values to slice and dice business transaction data later on. You must provide unique fields for each milestone. Validation fails if you create more than one milestone with the same set of fields or with the same set of mandatory and optional filters.

For each milestone, a field called `milestoneReached` is added to the associated event. The value is true if the milestone has been reached, otherwise it is null. You can use this field to query all events for which a milestone has been reached. You can also use `milestoneReached` to build a funnel widget that shows dropoffs for each milestone. The milestone name is appended to the field name as follows:

- `milestone1_name.milestoneReached`
- `milestone2_name.milestoneReached`

Likewise, the default fields and filters are also appended with and namespaced by milestone name.

At the Business Journey level, a field named `completed` tracks the status of the workflow using boolean values. True represents a completed Business Journey, whereas False indicates that the journey is in progress or is incomplete. Use the Analytics Search UI or an ADQL query for the desired Business Journey with the query string `completed` to view the status of the associated events.

The end-to-end time of the milestones comprising the Business Journey is represented by the `totalTime` field. This field will be empty when the Business Journey is partially complete. As each milestone completes and the event data is captured in the Business Journey event, the total time is updated.

Mandatory filters, mandatory fields, and fields extracted by default depend on the event type as shown in the following table.

<table>
<thead>
<tr>
<th>Event Type</th>
<th>Mandatory Filters</th>
<th>Mandatory Fields</th>
<th>Fields extracted by Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transactions</td>
<td>Application Name, Business Transaction Name, Tier Name</td>
<td>Primary Key</td>
<td>eventTimestamp, requestGuid, requestExperience</td>
</tr>
<tr>
<td>Logs</td>
<td>Source Type</td>
<td>Primary Key</td>
<td>eventTimestamp, logLevel, logType</td>
</tr>
<tr>
<td>Browser RUM</td>
<td>AppKey</td>
<td>Page Name</td>
<td>Page Type</td>
</tr>
<tr>
<td>------------</td>
<td>--------</td>
<td>-----------</td>
<td>----------</td>
</tr>
<tr>
<td>Mobile RUM</td>
<td>AppKey</td>
<td>Network Request Name</td>
<td>Primary Key</td>
</tr>
<tr>
<td>Custom Events</td>
<td>Source Type</td>
<td>none</td>
<td></td>
</tr>
</tbody>
</table>
Configure Business Journeys

On this page:

- Before You Begin
- Create a Business Journeys Definition
- Business Journeys Life Cycle

Related pages:

- View Business Journeys
- Create an Example Business Journey

This section describes data prerequisites, requirements, validation rules, limitations, and actions related to creating Business Journeys.

**Before You Begin**

Before creating Business Journeys ensure that the feature is enabled in your environment. You also need to review data prerequisites, validation rules for both names and Business Journey definition, and general limitations.

**Enable Business Journeys**

Business Journeys is enabled by default on SaaS environments. But on on-premises installations, the feature is disabled. To enable Business Journeys, an Administrator needs to configure certain properties in the Controller Settings.

Before starting the Event Service:

1. Open the `events-service-api-store.properties` file.
2. Change `ad.bizoutcome.enabled=false` to `true`.

Business Journey page displays a flow map detailing the business workflow across milestones. You can hide the flow map by changing the value of the `analytics.business.outcomes.flowmap.enabled` property to false in the Controller Settings.

**Enable Flow Maps**

As you create milestones, a flow map detailing the business workflow is automatically displayed in the Business Journey page. Additionally, selecting a Business Journey opens it in a new window with the flow map enabled. This behavior is controlled by the `analytics.business.outcomes.flowmap.enabled` property in the Controller Settings. Setting its value to true indicates that flow map is enabled for Business Journeys.

**Data Prerequisites for Defining Business Journeys**

Analytics creates Business Journey composite events by collecting events and fields that you have already captured as analytics data. In addition to built-in analytics data sources, custom event data can also be used to create Business Journeys. As you define the milestones in the Business Journey, you specify which events to use and which fields to extract from the events. You can extract fields collected by default or custom fields that you have configured for collection. You can add milestones with custom events created from Analytics API Keys.

You need to know your starting and ending event and the clearly defined steps in between. To collect additional fields in the events being captured by Application Analytics use one or more of the following techniques:

- Analytics Custom Events Data
- Data Collectors
- Collect Business Data From SQL Calls
- Configure Log Analytics Using Source Rules

**Restrictions and Caveats**

- Business Journey names are case-insensitive and stored as all lowercase. This means that two definitions named `application` and `Application`, for example, are considered a duplicate.
- The event type drop-down displays the lowercase definition name.
• Use the lowercase name for advanced queries.
• The primary key must be unique and not null.
  If the primary key holds null values, the underlying milestone events will not be stitched together to form a Business Journey event.
• Choose the primary key carefully, in particular, the cardinality of the values. The primary key is expected to uniquely identify and join milestone events together to form a meaningful business workflow. If the milestone events are misidentified by the primary key, inaccurate milestone events will be joined to produce faulty results.
• The primary key name cannot be changed unless the Business Journey definition is in the draft state.
• The Business Journey definition name cannot be changed after the definition is deployed. You can rename it while the definition is in the draft state.
• If a field related to a custom event is renamed, update the Business Journey definition created from the custom event for the renamed field to be correctly extracted.

Validation Rules

This section describes certain rules to validate Business Journey definition and naming convention for Business Journeys, milestones and extracted fields. You can save your work as a draft without invoking validation. When you are ready, you can use Validate and Save to check that your definition is acceptable. Saving the Business Journey definition does not enable the Business Journey. Data is not captured until you actually enable the Business Journey. Also, see Business Journeys Life Cycle.

Validation Rules for Names

This section describes the validation rules for naming Business Journeys, milestones, and extracted fields. Validation rules are not applied when you save your Business Journey definition as a draft. Validation is only invoked when you use Validate and Save. The validation rules include the following:

• Names of Business Journeys, milestones, and extracted fields must have an alphanumeric string containing a-z, A-Z and 0-9. The only special character allowed is the underscore "_".
• Special characters such as spaces, hyphens, dashes are not allowed in the names.
• Primary Key fields must have the same name (label) in all milestones in a Business Journey definition. The field name itself may vary in different event types.
• The primary key name cannot be changed unless the Business Journey definition is in the draft state. See Business Journeys Life Cycle for more information.
• The following reserved field names cannot be used to name milestones or fields in your Business Journey definition.
  • pickupTimestamp
  • eventTimestamp
  • totalTime
  • userExperience

Validation Rules for Business Journey Definition

A Business Journey definition must contain:

• A minimum of two milestones.
• A unique set of filters. More than one milestone with the exact same set of required and optional filters is not allowed in the same definition.
• Unique milestone names.
• Extracted field names must be unique (except the Primary Key field).
• There must be one Primary Key field per milestone.
• Slow threshold duration must be greater than 30 minutes.

Create a Business Journeys Definition

1. From the Controller UI, access Analytics > Business Journeys.
   A list of existing Business Journeys (if any) displays.
2. Click +/-Add and start defining the milestones in your business workflow.

3. To define milestones, click **Add Milestone**.

4. Type a name.

5. From the **Type** drop-down, select the source of the analytics data, such as Transactions, Logs, Browser Requests, Mobile Requests, and custom events.
6. For each data source, specify values for the mandatory filters that identify the data you want to capture. In the case of Transactions, specify the Application, Tier, and Business Transaction for each milestone you intend to create. In analytics, Tier represents a Java Virtual Machine (JVM) service, such as an authentication service, in your application environment. Business Transaction is the cross-tier processing path representing the request for a service provided by the application.

7. Define the milestones that comprise your workflow and the fields to extract at each milestone. You can reorder the milestones in the UI using drag-and-drop. Once you name the primary key, you can not change the name unless the definition is in the draft state. See Business Journey Life Cycle for more information.

   The primary key field is not populated for subsequent milestones because the primary key might have different field names in the event type for different milestones.

   Fields that are extracted by default by Business Journeys are not shown. Refer to the table describing Filters and Fields.

8. Save the definition using one of the following actions:
   - **Save As Draft**: saves the definition in draft form. No validation is performed and you can return to the definition to complete it at a later time.
   - **Validate and Save**: Performs necessary validation. Use this option when you have completed all milestones. The definition is saved, but is not enabled.

9. On the Health Thresholds tab, you can specify the values that determine acceptable performance. See ‘Business Journeys Health Thresholds’ on Business Journeys for more details on how the user experience value is calculated. When you finish creating the Business Journey definition, you are ready to enable it.

10. Select it from the Business Journeys list and click Enable. The option is grayed out until you select a valid draft Business Journey from the list. To enable a Business Journey, the state must be "Valid Draft".

### Business Journeys Life Cycle

This section provides detail on each phase of Business Journeys life cycle.

**Draft**

You might not have all the required details at the beginning and therefore, creating the definition may involve multiple temporary versions. Use **Save As Draft** to capture your initial, incomplete definition and place your Business Journey into a Draft state.

- No analytics data is processed and no Business Journey events are created in draft state, so there is nothing to query.
- The primary key can be changed when the Business Journey is in draft state.
- The draft state is less restrictive with respect to validations and supports most update operations.

**Valid Draft**

Once the definition has the required details, use **Validate and Save**, which transitions the definition into the Valid Draft state. You can still modify the definition in this state. Business Journey events are not captured until you enable the definition.

**Enabled**
When you are ready to process data, use the **Enable** action. This transitions the definition to the Enabled state. Most definitions will spend their lifetime in the Enabled state.

- Configuration changes are synced to the Analytics Servers every two minutes. Therefore, once you enable a Business Journey definition, there could be a delay of up to two minutes for events to be generated.
- Incoming events are processed, composite Business Journey events are created and stored. You can query for the Business Journey events.
- You can modify the definition in the following ways:
  - Add, rename, or delete fields for extraction (non-default fields)
  - Add, rename, or delete milestones

**User Disabled**

At some point, you may want to disable the Business Journey. The **Disable** action transitions the Business Journey definition to the User Disabled state. In this state, no incoming events are processed. Existing events can still be queried.

You can update the definition in the following ways:

- Add, rename or delete fields for extraction (non-default fields)
- Add, rename or delete milestones

**Deleted**

When you no longer need the Business Journey definition, use the **Delete** action. This is the final phase of the definition's lifecycle. In this state, no incoming events are processed for it. Existing Business Journey events are not available for query.
View Business Journeys

Search for Business Journeys

Use the Analytics Search button in the Business Journeys UI for one-click access to the underlying events. Selecting an enabled Business Journey and clicking Analytics Search takes you directly to the Searches page in Basic mode. Analytics Search button is unavailable for definitions in Draft and Valid Draft states.

Alternatively, access the Analytics Search UI and use the event type drop-down to find your Business Journey event. The Business Journey name identifies the event type in the Search UI similar to custom events. In this example, it is “loanapprovals”.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>State</th>
<th>Created By</th>
<th>Created At</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdCapitalEvents</td>
<td>ADCapital Loans</td>
<td>Enabled</td>
<td>ec2-user</td>
<td>02/22/18 10:50:57 AM</td>
</tr>
<tr>
<td>ADFinanceEvents</td>
<td>AD Trader app</td>
<td>Enabled</td>
<td>ec2-user</td>
<td>02/22/18 10:32:12 AM</td>
</tr>
<tr>
<td>loanapprovals</td>
<td>Loan Approval Workflow</td>
<td>Enabled</td>
<td>ec2-user</td>
<td>02/26/18 12:30:16 PM</td>
</tr>
</tbody>
</table>
You can search the Business Journey data and create visualizations in the same fashion as for any other analytics event type.

Business Journeys also gives you the ability to perform analytics, slice and dice data, and create widgets. For example, using the loan approval Business Journey, visualize percentile values of approval time, loan amount by type, and loan applications by type as follows:

The 50th Percentile line indicates that 50% of the loan approvals occurred below this line. Similarly, each percentile line represents the corresponding percentage of loans approved below that line.

**Business Journeys ADQL Queries**

To use ADQL to query Business Journey events, you need to use the full namespaceied field name for each milestone. The field names are constructed by appending the milestone name to the field. When constructing queries with ADQL in the Search UI, you must use the full name of the field. The following image shows constructing a sample ADQL query.
A sample ADQL query for the loanapproval Business Journey is:

```sql
SELECT * FROM loanapprovals WHERE 
  eventTimestamp 
  userExperience 
  loanId 
  LoanAmount 
  LoanType 
  isCreditApproved 
  Submit.eventTimestamp 
  Submit.requestExperience 
  Submit.requestGUID 
  CreditCheck.eventTimestamp 
  CreditCheck.requestExperience 
  CreditCheck.requestGUID 
 CreditCheck.milestoneReached 
  Underwrite.eventTimestamp 
  Underwrite.requestExperience 
  Underwrite.requestGUID 
  Underwrite.milestoneReached
```

Business Journeys Metrics

Application Analytics generates out-of-the-box metrics at the Business Journey definition and milestone levels. You can view these metrics in the Metric Browser. The metric values are reported per minute. For example, an observed value of 5 indicates that it’s the count of the records received in the one-minute interval. If the number of records reported in the last five minute is 25, the Metric Browser reflects an approximate observed value, that is 5.

To calculate metrics, Analytics adds two fields named `timeTaken` and `totalTime`. The `timeTaken` field is added to the associated event for each milestone. This field represents the time taken for an event to reach a milestone from the previous milestone, and therefore, the first milestone will not have this value.

The `totalTime` field is added to the associated event for each Business Journey definition. `totalTime` represents the end-to-end time for an event to cover all the milestones comprising the Business Journey. It is the time an event takes to reach the last milestone fr
om the first one. For some reason, if the first or the last milestone is missing, \( \text{totalTime} \) will not be calculated for that event. For example, if the loan application is rejected in the CreditCheck stage, the workflow never reaches the approval stage. In this case, \( \text{totalTime} \) is not determined.

By using the above-mentioned fields, the following out-of-the-box metrics are generated.

### Milestone Level

**Average timeTaken:** Average of the values measured for the \( \text{timeTaken} \) field. Evaluates the events that started in the last one day and reached the milestone in the last one minute.

**Calls per minute:** Number of events that started within the last one day and reached the selected milestone in the last one minute.

### Definition level

**Average totalTime:** Average of the values measured for the \( \text{totalTime} \) field. Evaluates the events that started in the last one day and reached the first milestone in the last one minute.

**Calls per minute:** Number of Business Journey events measured per minute. This number is equal to the calls per minute value of the first milestone. The value is calculated by evaluating the events that started in the last one day and reached the first milestone in the last one minute.

The lookback period to generate out-of-the-box metrics is one day. The values for the metrics might not be accurate if events take more than a day to complete or to even reach the next milestone. This is due to the lack of sufficient event records for Analytics to generate correct values, leading to displaying incorrect values. For example, the average \( \text{totalTime} \) metrics will only be accurate if events finish within the lookback period, that is one day.

Note that the existing definitions will not have the out-of-the-box metrics created. You need to create new definitions to generate corresponding out-of-the-box metrics.

### Business Journeys Dashboard

Analytics provides two Views for Business Journeys: Card View and Grid View. Business Journeys page defaults to the Grid View, but after switching to the Card View the UI makes it as the preference. The next time you log in, Business Journeys page defaults to the Card View. Use the View button on the toolbar to switch between the Views.

The Card View displays an out-of-the-box dashboard for each Business Journey definition along with key metrics. The dashboard highlights the milestones associated with each Business Journey definitions as well as the following metrics:

- Percentage of events reached the last milestone from the first one
- Average total time an event takes to complete the Business Journey
- Number of events reached the milestone
- Average time an event takes to reach a milestone from the previous one

Dashboards for Business Journeys created prior to the release of Analytics version 4.5 do not display the **Average time to**
Time range menu determines what events are considered for calculating metrics. The events that are considered for calculating metrics are only those started in the selected time range.

- **End-to-End**: The average end-to-end time an event takes to complete the corresponding Business Journey. It is the average time taken by an event to reach the last milestone from the first one.
- **Conversion**: The conversion rate represents the percentage of events that reached the last milestone from the first one. It is calculated as \( \frac{(events \ reached \ last \ milestone) \times 100}{(events \ reached \ first \ milestone)} \).
- **Average time to reach the next milestone**: It is the average time an event takes to reach the next milestone from the current one. This value is represented by the number shown in between each milestone.
- **Event count per milestone**: It is the number of events that reached a particular milestone. This value is indicated by the number shown inside the circles representing each milestone.

To workaround, edit the Extract Field section of one of the milestones and add the new field or remove an existing field. After the definition is saved, Analytics starts generating the missing values. Within a few minutes, you should see values for **Average time to reach the next milestone** for the Business Journey you edited.
This section uses a loan application, AD-Capital, to show the workflow and several milestones or steps involved in processing a loan. Similar to a typical loan application process, the workflow in this Business Journey starts with application submission, followed by document verification, credit check, underwriting, and finally, loan approval.

About the Milestones Data

Data for each milestone comes from different applications and different AppDynamics event types, such as business transactions, logs, or end-user events.

In this workflow, the data associated with application submission comes from business transaction events, whereas document verification status comes from logs. Credit check and underwriting are performed by third-party service providers and the status is updated in logs. The loan approval status is then updated in the transaction events.

Extract Fields: Primary Key and Additional Fields

Choose a distinct primary key that uniquely identifies and ties together these independent milestones to represent loan application. In this example, loanId is used as the primary key. You enter loanId in the Primary Key field in each milestone.

In addition to the default information collected by the Business Journey, such as event timestamp, you can extract additional business information, such as customer details, loan amount, loan type, bank name, and so on using the optional fields. These additional fields provide further context for your Business Journey definition. You can also use these fields to run ADQL queries later on.

Define Milestones

In this example, the application submission milestone is defined. The mandatory filters to limit the events for this milestone are Type, Application, Tier, and Business Transaction. You can see that loanId is used as the primary key. Additionally, fields such as accountid, loanType, email, and so on are extracted in this milestone.
Similarly, the milestone for underwriting is defined as shown below:

Underwriting is performed by third-party service providers and the status is updated in logs. Therefore, Logs is used as the Type of event source.

Similarly, define all other milestones. Validate and save the Business Journey. It takes a few minutes to display the generated records from the associated events.

Troubleshoot the Example Business Journey

If you think the combination of fields and filters that are selected for each milestone is incorrect, you can validate it with an ADQL query.
search or by using the Analytics Searches widget as shown below:

The Search page shows all the valid values for your selection. On the screen, you see the valid Tiers and associated Business Transactions for AD-Capital.

Alternatively, you can visit the Tiers & Nodes page of the AD-Capital application in the Controller.

Run Analytics on the Example Business Journey
Use one of the search methods from the Searches screen to view data, run analytics, or visualize the performance of the Business Journey you have just created. In this example, the progress of the credit check process is displayed for different types of loans defined in AD-Capital.

View the Example Business Journey

You can view dashboards and widgets of the loan approval Business Journey on the Dashboards & Reports UI. The visualization makes it easy to understand the performance of loan application workflow.

The funnel widget is given on the left side of the image. This widget represents the overall conversion rate, the health of participating events, and the number of abandoners at each critical step in the loan application process. At the center of the dashboard is a widget
that illustrates slow business transactions at each step in the Business Journey. The widget on the lower right side of the image compares credit check providers on the basis of credit approvals. The widget on the upper right side displays the number of loans approved in different cities.
Experience Level Management

On this page:

- Working with XLM
- Configuration Overview
- XLM Exclusion Periods
- XLM Auditing

Related pages:

- View and Export XLM Compliance Data
- Configure Experience Level Management
- Migrate XLM Configurations Between Environments

The performance of business transactions and other types of events can affect user experience. Different levels of performance may correspond to service level agreements (SLAs), compliance targets, or compliance policies that your application must satisfy.

For example, an e-retailer might want to ensure that when customers add items to their carts, the transactions always complete within 100 milliseconds.

Experience Level Management (XLM) enables you to define experience levels. XLM shows you how your application is performing relative to those levels, in custom dashboards and in reports that you can view and export.

Working with XLM

Whether you speak of an experience level or a service level depends on whether your focus is user experience or SLAs, but you configure both through the same XLM UI. What this section says about experience levels applies equally to service levels.

You can define experience levels for any kind of data that meets the following criteria:

1. The data must be numerical.
2. The data must be individual (per event) values, not aggregates.

Since data collectors provide individual values, they are a good basis for experience levels as long as the values are numeric. By contrast, metrics and information points cannot be used as the basis for experience levels, because they provide aggregate values.

The following are all good bases for defining experience levels:

- Sales in dollars per week during the time an ad campaign is running
- Application response times calculated from business transaction events
- End-user response times calculated from RUM and synthetic events
- Custom analytics such as login times for platinum customers or item checkout response times for London customers

Compliance against the configured thresholds is calculated in daily intervals from the specified start date. The local time zone starting time for each day is converted to midnight of the equivalent Greenwich Mean Time (GMT) date. For example, 07/30 12:00 am GMT on 07/30. The reporting job that calculates the XLM data runs every day at midnight and noon GMT. The results are updated the next time the reporting job runs.

XLM aggregates the results of reporting jobs into weekly or monthly periods, according to your choice of Compliance Period settings, including Time Zone.

When viewing XLM data for a week or month, you can drill down into daily granularity. XLM also allows you to

- Create XLM reports in the time zone of your choice
- Export reports of however many reporting periods you choose, in CSV format
- Add your XLM report to a custom dashboard for periodic generation and delivery

You can export XLM configurations, and migrate them from one environment to another. You can also view and export the XLM Audit Trail, which automatically records XLM configurations and changes to them.

If you revise the configuration, the new configuration takes effect the next time the XLM reporting job runs. Past data is immutable and configuration changes do not affect the values.
To see which configuration was in effect at a particular moment in a reporting period, view the XLM Audit Trail.

**Configuration Overview**

In the XLM UI, you configure Properties, Compliance Target Settings, and optional Exclusion Periods.

- **Properties** specify what type of event to measure and what filters to apply.
- **Compliance Target Settings** specify the performance criteria you want XLM to report, through
  - the **Compliance Target**, which defines the desired level of performance for the properties
  - **Daily Target Thresholds** which define Normal, Warning, and Critical performance as percentages of the compliance target
  - **Treat Errors as Critical**, an option helpful for applications where apparently normal performance thresholds tend to conceal errors
- **Compliance Target Settings** also specify how you want your XLM reports structured in terms of time, through
  - the **Start Date**—the date you want the compliance calculation to begin
  - the **Compliance Period**, which
    - you can think of as the reporting period
    - can be weekly or monthly; choose one or the other according to your SLA requirements
  - the **Time Zone** of your choice
- **Exclusion Periods** help you account for performance deviations caused by upgrades, maintenance, or usage patterns associated with weekends or holidays.

Here is how our e-retailer example might look:

- The properties have **Transactions** as the event type, filtered to exclude all but an **Add to Cart** business transaction, for the relevant application
- The compliance target is **Response Time (ms)**, set to 100
- Values for daily target thresholds are 95% to 100% for Normal, 90% to 95% for Warning, and 0% to 90% for Critical
- The compliance period is **Weekly**
- The time zone is **GMT+02:00** for Amsterdam, where our hypothetical e-retailer has customers

Configured this way, XLM reports that performance is **Normal** for each week-long reporting period when the Add to Cart business transaction completes within 100 milliseconds greater than 95% of the time.

For periods when Add to Cart performance meets the target 95% of the time or less, but above 90% of the time, performance is at **Warning** level.

For periods when performance meets the target 90% of the time or less, performance is **Critical**.

This example configuration is illustrated below.
XLM Exclusion Periods

Exclusion periods are time periods which you want to exclude from an XLM report.

For example, scheduled upgrades and maintenance can disrupt performance, so you may not want those periods reflected in XLM reports. Likewise, you may want XLM reports to exclude weekends, holidays, or other times with atypical usage patterns.

XLM queries executed against your Analytics event data observe only the exclusion periods you explicitly configure in the XLM UI. You configure exclusion periods using the time zone of your choice. Under the hood, AppDynamics keeps track of the exclusion period in the corresponding GMT dates.

XLM does not recognize exclusion and maintenance periods that you configure at the Controller level.
XLM Auditing

All changes to XLM configurations produce an immutable audit record called an XLM Audit Trail, which contains complete XLM configuration information. Since XLM audit reporting is separate from all other reporting in the AppDynamics Controller UI, the XLM Audit Trail is distinct from other available reports, such as the Controller Audit Report. You can export the XLM Audit Trail to a CSV file.

Each XLM configuration is limited to 1000 updates. If that limit is reached, the oldest audit records are dropped as new ones are added.

The XLM audit records contain information about what operations are performed and what fields are changed. The UI presents XLM audit records as a sequential timeline.
Configure Experience Level Management

On this page:
- Configure an XLM report
- Treat Errors as Critical Failures

Related pages:
- View and Export XLM Compliance Data
- Migrate XLM Configurations Between Environments

To configure Experience Level Management (XLM), you specify Properties, Compliance Target Settings, and optional Exclusion Periods as described earlier.

Configure an XLM report

1. From the Controller UI, choose Analytics > Experience Levels and click Add.

2. In the Properties section, name your XLM report uniquely. Select the event type and any filters necessary for isolating the data that you want to evaluate.

3. Select the desired field, target value, and daily target thresholds. Choose a time zone, report start date, and reporting period. Past dates are not allowed.
4. (Optional) Select Treat Errors as Critical Failures.
5. (Optional) Specify Exclusion Periods using the time period of your choice to denote periods during which data should be not collected.

6. Save the configuration. Data is collected and compliance is calculated in an automatic job that runs twice per day.

Treat Errors as Critical Failures

You can choose to treat events with errors as critical failures.

This enables you to identify instances where an event meets the compliance target but has a user experience status of Error.

For example, if you have defined Response Time of under 2000 milliseconds as the compliance target, an event with a response time of 100 milliseconds meets the compliance target, even if that event finished prematurely because of an error. In that case, XLM reporting that performance as Normal could be considered a false positive.

When you enable Treat Errors as Critical Failures, XLM will catch instances like this. Whether this option is optimal for your application is a subjective judgment for you to make.

Treat Errors as Critical Failures is only supported for event types that contain a user experience or similar field, as shown in the table below.

<table>
<thead>
<tr>
<th>Event Type</th>
<th>Experience Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transaction Event</td>
<td>userExperience</td>
</tr>
<tr>
<td>Browser Record</td>
<td>pageexperience</td>
</tr>
<tr>
<td>Mobile Snapshot (Network Request)</td>
<td>networkrequestexperience</td>
</tr>
<tr>
<td>Browser Session</td>
<td>pageexperience</td>
</tr>
<tr>
<td>Mobile Session</td>
<td>networkrequest.networkrequestexperience</td>
</tr>
</tbody>
</table>
View and Export XLM Compliance Data

On this page:
- Viewing XLM Data
- Adding XLM Reports to Custom Dashboards
- Exporting XLM Data

Related pages:
- Configure Experience Level Management
- Migrate XLM Configurations Between Environments

In the AppDynamics Controller UI, click **Analytics > Experience Level Management** when you want to view or export XLM data.

**Viewing XLM Data**

You can view the last five reporting periods for your XLM data in a card or list view. To access data for earlier reporting periods, export the data as described below.

Timelines show dates in your local timezone and indicate how local time relates to Greenwich Mean Time. For example, if your local timezone is Pacific Standard Time, **Date (GMT-7:00)** appears beneath timelines.

Both views enable you to drill down from multiple reporting periods, to days, to individual events.

**Card View**

Experience Level Management
Click any reporting period for a daily breakdown.

Right-click any day to view associated events.

This brings you to an Analytics > Searches page where you can inspect the entire day's events.

**List View**

As in the card view, you can:

- Click any reporting period for a daily breakdown
- Right-click any day to view associated events
Adding XLM Reports to Custom Dashboards

You can add XLM reports as read-only widgets to your custom dashboards in two ways:

- One at a time, from the Actions drop-down, or
- One or more at once, from Add to Dashboard in the Experience Level Management window as shown below.

From the Experience Level Management window, click Add to Dashboard. Alternatively, click the Actions drop-down on an individual XLM report.

### Experience Level Management

<table>
<thead>
<tr>
<th>Name</th>
<th>Current Period Average</th>
<th>Timeline</th>
<th>Time Zone</th>
<th>Created By</th>
<th>Last Modified On</th>
<th>Last Modified By</th>
</tr>
</thead>
<tbody>
<tr>
<td>XLM_DifferentTimezone_1</td>
<td>20.62%</td>
<td>06/11</td>
<td>06/18</td>
<td>GMT-07:00</td>
<td>06/24/18 11:52:43 PM</td>
<td></td>
</tr>
<tr>
<td>XLM_Transactions_1</td>
<td>95.38%</td>
<td>06/11</td>
<td>06/18</td>
<td>GMT-06:30</td>
<td>06/24/18 11:48:59 PM</td>
<td></td>
</tr>
<tr>
<td>XLM_BrowserRequests_1</td>
<td>1.69%</td>
<td>06/11</td>
<td>06/18</td>
<td>GMT-05:30</td>
<td>06/24/18 11:48:10 PM</td>
<td></td>
</tr>
<tr>
<td>test2_xlm_exitualion</td>
<td>19.98%</td>
<td>06/03</td>
<td>06/16</td>
<td>GMT-07:00</td>
<td>06/23/18 3:21:13 PM</td>
<td></td>
</tr>
<tr>
<td>test1_xlm</td>
<td>91.19%</td>
<td>06/03</td>
<td>06/15</td>
<td>GMT+05:30</td>
<td>06/22/18 1:17:49 PM</td>
<td></td>
</tr>
</tbody>
</table>
Exporting XLM Data

You can export XLM data using the Actions drop-down menu, which also offers additional views and actions.

You can export as many reporting periods as you wish. The export format is CSV.

The XLM export file shows the latest configuration along with compliance data.
Migrate XLM Configurations Between Environments

Related pages:
- Configure Experience Level Management
- View and Export XLM Compliance Data

You can migrate XLM configurations from one environment to another. For example, you might first deploy a configuration to a staging or test environment and later migrate it to production.

In order to migrate an XLM configuration, you make two API calls.

1. Export XLM configuration(s) from one environment.
   
   GET /controller/analytics/xlm/configuration
   
   Optionally, supply a configuration ID, or a comma-delimited list of configuration IDs, as a query parameter.
   
   If this parameter is omitted, the API returns a bulk export of all XLM configurations.

2. Import a single XLM configuration into another environment.

   POST /controller/analytics/xlm/configuration
ADQL Reference

The AppDynamics Query Language (ADQL) is a query language for searching data available in Application Analytics. ADQL is similar to the SELECT statement in the widely used Structured Query Language (SQL) but is designed specifically for AppDynamics Analytics data. Like the SELECT command in SQL, ADQL enables you to specify the data source (such as transaction events, EUM events, or logs), a list of fields to retrieve, and conditions for selecting events from the analytics data store.

With ADQL, you can construct queries in the following environments:

- Analytics Search window in the Controller UI using Advanced Mode
- Analytics Query Events APIs, in the body of the query request

To simplify entering your queries in the Analytics Search Advanced mode UI, the ADQL "auto-complete" feature displays many keywords, a list of available fields, or functions based on the stage of your typed search. For details on using auto-complete, see Creating Advanced Searches for Analytics.
ADQL Typographical Conventions and Symbols

ADQL reference topics use custom typographical conventions and symbols to describe the syntax for ADQL queries.

<table>
<thead>
<tr>
<th>Convention</th>
<th>Name</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOLD</td>
<td>Bold typeface</td>
<td>Bold typeface indicates ADQL keywords and operators, such as SELECT, FROM, AND and so on.</td>
</tr>
<tr>
<td>field_name and numeric_field_name</td>
<td>N/A</td>
<td>The terms field_name and numeric_field_name represent field names that appear in your analytics data. Field names depend on the event type. See ADQL Data.</td>
</tr>
<tr>
<td>Monospace typeface</td>
<td>Monospace typeface</td>
<td>Function names, such as count are indicated by monospace typeface.</td>
</tr>
<tr>
<td>event_type</td>
<td>N/A</td>
<td>The term event_type represents the available event types collected from your applications. Valid values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• transactions</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• logs</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• browser requests</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• mobile requests</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• mobile crash reports</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• analytics custom events</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• synthetic sessions</td>
</tr>
<tr>
<td>Italic typeface</td>
<td>Italics</td>
<td>Italics represent variables or placeholders. You supply the actual value.</td>
</tr>
<tr>
<td>[ ]</td>
<td>Square brackets</td>
<td>Terms in square brackets are optional. Nested square brackets indicate that the term inside is optional and can only be included if the optional term in the outer square brackets is included.</td>
</tr>
<tr>
<td></td>
<td>Pipe</td>
<td>The pipe character separates alternate items to indicate one of the items may be included. You can read it as &quot;or&quot; in a syntax statement.</td>
</tr>
<tr>
<td>{ }</td>
<td>Curly braces</td>
<td>Curly braces are used to group items to disambiguate the definition of items.</td>
</tr>
<tr>
<td>( )</td>
<td>Parentheses</td>
<td>Parentheses are used to indicate the order of operation in an expression.</td>
</tr>
<tr>
<td>...</td>
<td>Ellipsis</td>
<td>An ellipsis indicates the preceding term may be repeated.</td>
</tr>
</tbody>
</table>
ADQL Queries

ADQL Reference:

ADQL Query Syntax

ADQL uses the SELECT statement combined with filtering statements to return sets of data, which can optionally be visualized and added to Custom Dashboards.

ADQL statements are referred to as queries. The query syntax is as follows:

```
SELECT * | [(DISTINCT) expression [AS alias] [, expression [AS alias]]...]  
FROM event_type  
[WHERE condition_expression]  
[SINCE timevalue [UNTIL timevalue]]  
[HAVING condition_expression]  
[ORDER BY {field_name | alias} [ASC | DESC] [, {field_name | alias} [ASC | DESC]]...]  
[LIMIT integer [, integer]...]
```

The SELECT clause and FROM clause are required. All other clauses are optional. Type spaces around each keyword.

For GROUP BY functionality, see GROUP BY.
SELECT Clause

ADQL Reference:

Overview

The ADQL event query begins with a SELECT clause, which can take one or more arguments.

```
SELECT * | [[DISTINCT] field_expression [AS alias] [, field_expression [AS alias]]...]
```

- A single asterisk * in the SELECT clause means return the list of fields for each event.
- To return specific fields for each event, specify a comma-separated list of fields.
  Optionally, each field_expression can be aliased to a label using the AS construct.
- Use the DISTINCT qualifier to return unique combinations of the specified field values. If no limit specified, the top 10 results by count are returned. See LIMIT Clause.

**DISTINCT** can not be applied to a wild card SELECT query, such as

```
SELECT DISTINCT *
```

- String values must be surrounded by either single quotes or double quotes, such as 'hello' and "hello".

The field_expression can take several forms. See ADQL Expressions for a description of the syntax and rules.

Examples

<table>
<thead>
<tr>
<th>Query</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT count(transactionName) FROM transactions WHERE application='yourAppName'</td>
<td>Returns a count of business transactions for the specified application.</td>
</tr>
<tr>
<td>SELECT * FROM logs WHERE sourceType='yourLogFile' AND message LIKE GET</td>
<td>Returns log events containing a specified keyword.</td>
</tr>
<tr>
<td>SELECT * FROM transactions WHERE application='yourAppName' AND transactionName='yourBTName'</td>
<td>Returns all fields for business transactions in the specified application matching the specified transaction name.</td>
</tr>
<tr>
<td>SELECT DISTINCT transactionName FROM transactions WHERE application='yourAppName'</td>
<td>Returns a list of unique transaction names from the specified application.</td>
</tr>
<tr>
<td>SELECT customerName AS &quot;First And Last Name&quot;, age AS &quot;Years Old&quot;, ...</td>
<td>Returns the specified fields relabeled as specified in the query statement. In this query, customerName and age are user-defined fields.</td>
</tr>
</tbody>
</table>
**AS Clause**

**ADQL Reference:**

**Overview**

Use the AS clause to rename a field to a label of your choice. The label is also referred to as an alias.

field_expression AS alias

Returns the field with the specified alias. The alias may be any valid identifier (a letter followed by any number of letter or digit characters) or a value in quotes. If the value of the alias contains spaces or special characters, it must be in quotes. For a list of special characters, see ADQL Expressions.

**Examples**

SELECT eventTimestamp AS 'Time Stamp' FROM transactions WHERE application=yourAppName
FROM Clause

Overview

Use the FROM clause to specify the event type. For example:

\[
\text{FROM event\_type}
\]

Event Types

Event types describe the source of the analytics data, such as business transactions (transactions) or browser requests. Each event type has a UI field name and an Events Service internal name. The Events Service internal name is used in ADQL queries.

The following table lists the event types with their UI names and Events Service internal names.

<table>
<thead>
<tr>
<th>Event Type UI Label</th>
<th>Events Service Internal Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transactions</td>
<td>transactions</td>
</tr>
<tr>
<td>Logs</td>
<td>logs</td>
</tr>
<tr>
<td>Browser Requests</td>
<td>browser_records</td>
</tr>
<tr>
<td>Browser Sessions</td>
<td>session_records</td>
</tr>
<tr>
<td>Mobile Requests</td>
<td>mobile_snapshots</td>
</tr>
<tr>
<td>Mobile Sessions</td>
<td>mobile_session_records</td>
</tr>
<tr>
<td>Mobile Crash Reports</td>
<td>mobile_crash_reports</td>
</tr>
<tr>
<td>Synthetic Sessions</td>
<td>synth_session_records</td>
</tr>
</tbody>
</table>

Examples

<table>
<thead>
<tr>
<th>For this result</th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return all fields for browser requests</td>
<td>\text{SELECT } * \text{ FROM browser_records WHERE appkey='E2E-AAB-AU M'}</td>
</tr>
<tr>
<td>Return all fields for the business transactions in your application</td>
<td>\text{SELECT } * \text{ FROM transactions WHERE application='yourAppName'}</td>
</tr>
<tr>
<td>Return all fields for the logs specified by &quot;yourLogFile&quot;</td>
<td>\text{SELECT } * \text{ FROM logs WHERE sourceType='yourLogFile'}</td>
</tr>
</tbody>
</table>
WHERE Clause

ADQL Reference:

Overview

Use the WHERE clause to specify one or more condition expressions separated by logical operators such as AND, OR.

The WHERE clause uses the following syntax:

[WHERE condition_expression]

See ADQL Expressions for a description of the condition expression syntax. Condition expressions use Comparison Operators.

Examples

To filter logs based on error or warning log level, use a query similar to the following:

SELECT * FROM logs WHERE sourceType='yourLogFile' AND (logLevel='ERROR' OR logLevel='WARN')
GROUP BY

Overview

ADQL does not have an explicit GROUP BY clause. However, queries containing field names in the SELECT clause behave as GROUP BY statements if the query contains metric functions or bucketing functions. The following examples illustrate how this works.

Group by queries are limited to containing up to five terms. Terms can be either fields (which use an implicit GROUP BY) or bucketing functions, such as series.

Aggregation Examples

In this example, the query groups the \texttt{avg(responseTime)} aggregation based on the non-aggregation field, application. So the result contains the average response time calculated for each application.

\texttt{SELECT application, avg(responseTime) FROM transactions}

Sample results:

By adding tier as a second non-aggregation field, the query groups the \texttt{avg(responseTime)} aggregation based on the non-aggregation fields, application, and tierName. Because there are multiple grouping fields, the second non-aggregation field, tier name, is sub-grouped under the application grouping. So each combination of application and tier name together forms a group over which the average response time is calculated.

Sample results:
For SQL users, the following two statements produce similar results in their respective environments:

- ADQL query: `SELECT application, segments.tier, avg(responseTime) FROM transactions`
- SQL query: `SELECT application, segments.tier, avg(responseTime) FROM transactions GROUP BY applicationName, segments.tier`

**Bucketing Function Example**

Bucketing functions such as `series` and `range` define groups over which other aggregations can be calculated. The buckets are specified by the parameters of the respective functions and form groups over which subsequent aggregations are calculated.

For example, the following query calculates the average response time over two-minute intervals (or buckets):

`SELECT series(eventTimestamp, '2m'), avg(responseTime)`

Sample results:
You can also combine an aggregation with a bucketing function and implicit GROUP BY. The following query returns the average response time grouped by tier over the series of six-hour intervals.

```
SELECT series(eventTimestamp, '6h'), avg(responseTime) FROM transactions WHERE application = 'ECommerce-B2E'
```

Sample Results:

<table>
<thead>
<tr>
<th>series(eventTimestamp)</th>
<th>avg(responseTime)</th>
</tr>
</thead>
<tbody>
<tr>
<td>08/22/16 3:40:00 PM</td>
<td>46.172</td>
</tr>
<tr>
<td>08/22/16 3:42:00 PM</td>
<td>56.328</td>
</tr>
<tr>
<td>08/22/16 3:44:00 PM</td>
<td>44.537</td>
</tr>
<tr>
<td>08/22/16 3:46:00 PM</td>
<td>50.068</td>
</tr>
<tr>
<td>08/22/16 3:48:00 PM</td>
<td>50.157</td>
</tr>
<tr>
<td>08/22/16 3:50:00 PM</td>
<td>43.767</td>
</tr>
<tr>
<td>08/22/16 3:52:00 PM</td>
<td>51.759</td>
</tr>
<tr>
<td>08/22/16 3:54:00 PM</td>
<td>37.514</td>
</tr>
</tbody>
</table>
```
SELECT eventTimestamp('6h'), segments.tier, avg(responseTime) FROM transactions
```

### Current Search Results

<table>
<thead>
<tr>
<th>Event Time</th>
<th>Tier</th>
<th>Avg Response Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>08/22/16 11:00:00 AM</td>
<td>ECommerce-Services</td>
<td>41.889</td>
</tr>
<tr>
<td>08/22/16 11:00:00 AM</td>
<td>Inventory-Services</td>
<td>251.444</td>
</tr>
<tr>
<td>08/22/16 11:00:00 AM</td>
<td>Order-Processing-Services</td>
<td>251.444</td>
</tr>
<tr>
<td>08/22/16 5:00:00 PM</td>
<td>Inventory-Services</td>
<td>4,009.542</td>
</tr>
<tr>
<td>08/22/16 5:00:00 PM</td>
<td>ECommerce-Services</td>
<td>588.622</td>
</tr>
<tr>
<td>08/22/16 5:00:00 PM</td>
<td>Order-Processing-Services</td>
<td>4,009.542</td>
</tr>
<tr>
<td>08/22/16 11:00:00 PM</td>
<td>ECommerce-Services</td>
<td>542.803</td>
</tr>
<tr>
<td>08/22/16 11:00:00 PM</td>
<td>Inventory-Services</td>
<td>5,665.215</td>
</tr>
<tr>
<td>08/22/16 11:00:00 PM</td>
<td>Order-Processing-Services</td>
<td>5,665.215</td>
</tr>
<tr>
<td>08/23/16 5:00:00 AM</td>
<td>ECommerce-Services</td>
<td>9.984</td>
</tr>
<tr>
<td>08/23/16 11:00:00 AM</td>
<td>ECommerce-Services</td>
<td>10.580</td>
</tr>
</tbody>
</table>
ORDER BY Clause

ADQL Reference:

Overview

The ORDER BY clause enables you to sort the query results by one or more fields. The syntax is:

ORDER BY {field_name | alias} [ASC | DESC] [, {field_name | `alias`} [ASC | DESC]]...

Rules governing ordering include:

- Sort order, ascending (ASC) or descending (DESC) is applied per field. If not specified, the ascending (ASC) order is the default.
- ORDER BY is supported on fields or aliases specified in the SELECT clause.
- Aliases used in the ORDER BY clause should be surrounded with back quotes if they contain spaces or other special characters.
- ORDER BY works on single-value metric functions and single-value math expressions. ORDER BY can not be used with percentile or stats.
- ORDER BY on nested fields or metric aggregations that operate on nested fields is not supported. A nested field is any field that has a `.` in the name, such as segments.errorList.errorType or btdata.estimatedtime.

Ordering functionality for metric and bucketing functions includes:

- ORDER BY can be used to order the buckets on a bucketed field in a bucketing function. For example, see the “Series aggregations · Ordering based on the keys of the series buckets.” in the table below.
- ORDER BY can be used on metric functions that follow bucketing functions, including filtered metric functions. Ordering by the metric value orders the resulting buckets. The metric function must be aliased and referenced by its alias such as:

  SELECT transactionName, avg(responseTime) AS averageResponseTime FROM transactions ORDER BY averageResponseTime

- Compound orderings can be specified for functions that provide implicit GROUP BY functionality.
- A single ordering can be defined for the series function, no ordering is allowed for the range function.
- ORDER BY can accept math expressions on fields, but not math expressions on metric functions. If the value of a math expression is null it is ordered at the end of the results in ascending order. The following is an invalid query:

  SELECT transactionName, (discountValue) / (cartTotal) AS ratio FROM transactions ORDER BY ratio

Examples

<table>
<thead>
<tr>
<th>For this result</th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>An ordering of queries with no aggregations</td>
<td>SELECT field_name FROM event_type ORDER BY field_name ASC</td>
</tr>
<tr>
<td>Group By - Ordering by the lexicographic ordering of the terms. The fields included in the ORDER BY clause must be in the same order as in the query itself.</td>
<td>SELECT field_name, count(*) FROM event_type ORDER BY field_name DESC</td>
</tr>
<tr>
<td>Returns a sorted set of timestamp ranges</td>
<td>SELECT series(eventTimestamp, 10m), count(*) FROM transactions ORDER BY eventTimestamp DESC</td>
</tr>
<tr>
<td>Series aggregations - Ordering based on the keys of the series buckets.</td>
<td>SELECT series(salary, 20000), count(salary) FROM event_type ORDER BY salary DESC</td>
</tr>
<tr>
<td>Date series aggregations - Ordering based on the date value of the keys of the series buckets</td>
<td><code>SELECT series(eventTimestamp, '1h'), count(eventTimestamp) FROM transactions ORDER BY eventTimestamp</code></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>Order by an alias.</td>
<td><code>SELECT responseTime AS RT FROM transactions ORDER BY RT</code></td>
</tr>
<tr>
<td>Order by an aliased math expression.</td>
<td><code>SELECT discountValue / cartTotal AS ratio FROM transactions ORDER BY ratio</code></td>
</tr>
</tbody>
</table>
HAVING Clause

Overview

Use the HAVING clause on the output produced by another aggregation. The HAVING clause operates on an expression to determine the intervals to be shown in the response. The metric specified in the expression must be numeric and the expression must return a boolean value for the HAVING clause to work.

Additionally, the HAVING clause must be referenced by an alias in the query as follows:

```
SELECT customerName, count(*) as Requests, avg(responseTime) as ResponseTime FROM transactions HAVING Requests > 10000
```

where Requests is the alias referenced in the query.

Syntax

```
SELECT selectItems
    FROM relation
    WHERE where=booleanExpression
    SINCE statement
    HAVING havingClause
    ORDER BY sortItems
    LIMIT limits
;
```

where havingClause is a boolean expression. The WHERE, ORDER BY, and LIMIT clauses and SINCE statement are optional.

Examples

Usage of HAVING clause with different filters is given below:

Simple Comparison Filter

```
SELECT transactionName, avg(responseTime) as AVRT FROM type HAVING AVRT > 0
```

Simple Range Filter
SELECT transactionName, avg(responseTime) as AVRT FROM type HAVING AVRT BETWEEN [10, 90]

**Simple List Filter**

SELECT transactionName, avg(responseTime) as AVRT FROM type HAVING AVRT IN (20, 30, 40, 60)

**Compound Filter**

SELECT transactionName, avg(responseTime) as AVRT FROM type HAVING AVRT > 20 AND AVRT < 90
SELECT transactionName, avg(responseTime) as AVRT FROM type HAVING AVRT < 20 OR AVRT > 90
SELECT transactionName, avg(responseTime) as AVRT FROM type HAVING AVRT > 20 AND AVRT < 90 AND AVRT IN (70, 80)

**Not Filter**

SELECT transactionName, avg(responseTime) as AVRT FROM type HAVING NOT AVRT > 50

**Not Compound Filter**

SELECT transactionName, avg(responseTime) as AVRT FROM type HAVING NOT (AVRT > 20 AND AVRT < 90)

**Multi-Metric Filter**
SELECT transactionName, avg(responseTime) as AVRT, max(responseTime) as MAXRT FROM type HAVING NOT (AVRT > 20 AND AVRT < 90) AND MAXRT > 130

Limitations

The **HAVING** clause is not supported for the following:

- Nested fields in Elasticsearch
- Multi-metric functions, such as `percentiles` and `stdev`
- Filtered metric functions (filter function)
- Invalid queries as given below

<table>
<thead>
<tr>
<th>Unsupported Conditions/Filters</th>
<th>Queries</th>
</tr>
</thead>
<tbody>
<tr>
<td>Like</td>
<td>SELECT transactionName, avg(responseTime) as AVRT FROM transactions HAVING AVRT LIKE '10'</td>
</tr>
<tr>
<td>Exists</td>
<td>SELECT transactionName, avg(responseTime) as AVRT FROM transactions HAVING AVRT IS NOT NULL</td>
</tr>
<tr>
<td>Not Aliased</td>
<td>SELECT avg(responseTime) as avtr FROM type HAVING avg(responseTime) &gt; 10</td>
</tr>
<tr>
<td></td>
<td>SELECT txName, count(<em>), node, count(</em>) FROM transactions HAVING count(*) &gt; 10</td>
</tr>
<tr>
<td>Nested</td>
<td>SELECT transactionId, AVG(segments.numCalls) as av_numCalls, MIN(segments.numCalls), MAX(segments.numCalls), SUM(segments.numCalls) FROM transactions HAVING av_numCalls &gt; 2</td>
</tr>
<tr>
<td>Filter Function</td>
<td>SELECT appraisalrating, filter(min(salary), where salary &gt; 60000) as val FROM transactions HAVING val &gt; 60000</td>
</tr>
<tr>
<td>Multi-Valued Metric Functions</td>
<td>Percentile: SELECT appraisalrating, percentile(salary, 94) as per_sal FROM transactions HAVING per_sal &gt; 2 ORDER By appraisalrating</td>
</tr>
<tr>
<td></td>
<td>Standard Deviation: SELECT appraisalrating, stdev(salary) as stdev_sal FROM transactions HAVING stdev_sal &gt; 2 ORDER By appraisalrating</td>
</tr>
<tr>
<td>Missing Parent Aggregation</td>
<td>SELECT avg(responseTime) as AVRT FROM transactions HAVING AVRT &gt; 2</td>
</tr>
</tbody>
</table>
SINCE...UNTIL Clause

**ADQL Reference:**

Use the SINCE...UNTIL clause on a time range in a query. If the UNTIL portion is omitted, the query matches all the events that occurred after the time specified in the SINCE portion. A query can contain both a SINCE...UNTIL clause and a WHERE clause. In such cases, WHERE argument comes before SINCE...UNTIL and is applied in that order. The time range specified in the clause overrides any other time limits applied to the query, including the time range selection in the UI.

**Syntax**

```
SINCE timevalue [UNTIL timevalue]
```

where *timevalue* is a combination of an integer and (optionally) a time unit. If *timevalue* is omitted the value is interpreted as a Unix epoch timestamp in milliseconds.

<table>
<thead>
<tr>
<th>Unit Symbols</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[m</td>
<td>minute</td>
</tr>
<tr>
<td>[h</td>
<td>hour</td>
</tr>
<tr>
<td>[d</td>
<td>day</td>
</tr>
<tr>
<td>[w</td>
<td>week</td>
</tr>
</tbody>
</table>

**Examples**

<table>
<thead>
<tr>
<th>Query</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT * FROM transactions SINCE 1 h UNTIL 15 m</td>
<td>All transactions that occurred between 1 hour and 15 minutes ago</td>
</tr>
<tr>
<td>SELECT * FROM transactions SINCE 1 h</td>
<td>All transactions that occurred after one hour ago</td>
</tr>
<tr>
<td>SELECT * FROM transactions SINCE 1502147143000 UNTIL 1502148043000</td>
<td>All transactions that occurred between the Unix timestamps 1502147143000 and 1502148043000</td>
</tr>
</tbody>
</table>
LIMIT Clause

ADQL Reference:

Overview

The LIMIT clause enforces a limit on the returned search results. LIMIT works differently for non-aggregation and aggregation queries. The syntax is as follows:

[LIMIT integer [, integer]...]

An integer is a sequence of digits. If ORDER BY is not specified in the query, the results sort in descending timestamp order. The Analytics Search UI caps the number of search results at 10000 records.

Non-Aggregation Queries

Specify LIMIT N for non-aggregation queries to return the first N documents.

SELECT * FROM transactions LIMIT 10

For non-aggregation queries, using the Analytics Query API, can return a maximum of 10,000 results. If a limit value higher than 10,000 is specified, the value is overridden with the maximum value.

Aggregation Queries

For aggregation queries a list of values is allowed in the LIMIT clause. Each value applies to one "group by" term or bucketing function in the query. A "group by" term is a field name in an aggregation query. In the following example, field1 and field3 are group by terms, and series(field2, '1m') is a bucketing function:

SELECT field1, series(field2, '1m'), field3, count(*) ...

Bucketing functions consume a value from the LIMIT clause list of values, but that value is ignored because it doesn't apply to such functions. Because the default limit is 10, if there are more bucketing functions than user-specified limits, the LIMIT clause will use 10 buckets. For example, if there two limit values specified, but three functions or terms that could consume the limit values, the last one defaults to a limit of 10.

The maximum number of buckets is 1000 for the first group by term or bucketing function. Subsequent terms are limited to 100 buckets. A larger value is overridden and replaced with the maximums. Aggregation queries can return a maximum of 10000 results.

REST API usage for limits is slightly different. Refer to the section "Querying Events" in Analytics Events API.

Aggregation Examples

<table>
<thead>
<tr>
<th>Query</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT transactionName, avg(responseTime) FROM transactions</td>
<td>No limit specified, shows a maximum of top 10 results by count.</td>
</tr>
<tr>
<td>SELECT DISTINCT requestGUID FROM transactions</td>
<td>No limit specified, shows the top 10 results by count.</td>
</tr>
<tr>
<td>SELECT transactionName, avg(responseTime) FROM transactions LIMIT 50</td>
<td>Returns the top 50 results by count.</td>
</tr>
<tr>
<td>SELECT application, transactionName, avg(responseTime) FROM transactions LIMIT 500, 50</td>
<td>Returns the top 500 applications by count. Within each of those buckets, the top 50 transactionNames by count.</td>
</tr>
<tr>
<td>SQL Query</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| ```SQL
SELECT application, transactionName, avg(responseTime)
FROM transactions
LIMIT 2000, 500
``` | Because of the enforced limits, this query returns the top 1000 applications by count. Within each of those buckets, the top 100 transactionNames by count. |
| ```SQL
SELECT application, transactionName, avg(responseTime)
FROM transactions
LIMIT 20
``` | Returns the top 20 applications by count, within each of those buckets the top 10 (using the default) transactionNames by count.                                                                                   |
| ```SQL
SELECT application, series(eventTimestamp, '1m'), transactionName, avg(responseTime)
FROM transactions
LIMIT 20, 25, 30
``` | Returns the top 20 applications by count. Within those a date-time series with all the data. Within each date-time series bucket, shows the top 30 transactionNames by count. The value 25 is applied to the series function, but because series is a bucketing function, it does not actually use the limit. |
| ```SQL
SELECT application, series(eventTimestamp, '1m'), transactionName, avg(responseTime)
FROM transactions
LIMIT 20, 25
``` | Returns the top 20 applications by count. Within those, shows a date-time series with all the data. Within each date-time series bucket the top 10 (using the default) transactionNames by count. The value 25 is applied to the series function, but because series is a bucketing function, it does not actually use the limit. |
| ```SQL
SELECT series(eventTimestamp, '1m'), avg(responseTime)
FROM transactions
LIMIT 20
``` | LIMIT doesn’t apply to the functions, only to the group by terms, therefore in this query, the limit value has no effect and the query returns all the data in the series function. |
ADQL Expressions

Overview

ADQL uses two primary categories of expressions in ADQL queries:
- field expressions used in the `SELECT` clause
- condition expressions used in the `WHERE` clause

To build an expression, use field names, functions, numbers, and operators as indicated in the syntax described below.

Syntax of field_expression (SELECT clause)

- `{field_name | function | numeric_literal}
- `(expression)` - surround the expression with parentheses to denote the order of operation in the query
- math expressions such as:
  - unary_operator expression
  - expression binary_operator expression
- A condition_expression such as `[NOT]` expression `COMPARISON_OPERATOR` value can also be an alias used in a field_expression in the `SELECT` clause. For example:

```
SELECT field_name AS myField... WHERE myField > 2
```

Syntax of condition_expression (WHERE clause)

- `[NOT]` expression `COMPARISON_OPERATOR` value
- `(condition_expression)`
- `condition_expression` `LOGICAL_OPERATOR` `condition_expression`

The field expression syntax in the `SELECT` clause can also contain functions. The syntax of a function_expression is:

```
function(field_name, arguments, ...)
```

<table>
<thead>
<tr>
<th>Syntax element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>field_name or numeric_field_name</td>
<td>The name of a field in the specified event type. Field names require back quotes when the name contains spaces or special characters. For example, the field name, 'full name', is surrounded by back quotes because it contains a space. See the section &quot;Special Characters&quot; for the full list. Use the Events Service internal names in your ADQL queries. The list of default fields and their internal names are described under ADQL Data.</td>
</tr>
<tr>
<td>function</td>
<td>See Analytics Functions.</td>
</tr>
<tr>
<td>numeric_literal</td>
<td>A sequence of digits, optionally including a decimal point &quot;.&quot; or sign {+</td>
</tr>
<tr>
<td>unary_operator</td>
<td>The - <code>unary</code> operator negates the value of the operand.</td>
</tr>
<tr>
<td>binary_operator</td>
<td>Arithmetic operators: {*, /, %, +, -}. See Math Expressions.</td>
</tr>
<tr>
<td>COMPARISON_OPERATOR</td>
<td>A set of operators that compare values. See Comparison Operators for the full list.</td>
</tr>
<tr>
<td>LOGICAL_OPERATOR</td>
<td>See Logical Operators.</td>
</tr>
</tbody>
</table>
value

The syntax for a value = (numeric_literal | string | boolean). Values are text, numbers, dates or boolean values used to compare with the value in the specified field_name. The data type of the value must match the type of the specified field.

Text strings require quotes. Numbers, dates, boolean values (true or false) and null do not need quotes. Quotes can be either single or double quotes.

**Handling Fields Containing Spaces or Special Characters**

If the Events Service internal name contains a special character, then the name must be surrounded by back quotes when it is used in a field expression in the **WHERE** clause.

For a field name such as `segments.httpData.headers.User-Agent`, the ADQL query must use `segments.httpData.headers.'User-Agent'`. For example:

```
```

Field names added using data collectors are the most likely instances where spaces or other special characters might be used in naming.

**Special Characters**

Special characters in ADQL are the following:

- equals : '='
- not equals : '<>' or '!='
- less than : '<'
- less than or equals : '<='
- greater than : '>
- greater than or equals : '>='
- plus: '+'
- minus: '-'
- asterisk: '*'
- slash: '/'
- percent: '%'
- concatenation: '||'
- question mark: '?'
- open parens: '('
- open square bracket/brace: '['
- close parens: ')'
- close square bracket/brace: ']'
Analytics Functions

On this page:

- Overview
- Metric Functions
- Bucketing Functions
- String Functions
- Data Type Conversion Functions
- Using mathematical and now() functions on datetimes
- Functions That Modify Other Functions
- Combining Different Types of Functions

ADQL Reference:

Overview

Functions serve as arguments in the `SELECT` clause. Function types in ADQL include:

- Metric functions
- Bucketing functions
- String functions
- Data type conversion functions
- Functions that modify other functions

For many functions, only numeric fields are valid. "All" data types means `integer`, `floating point`, `datetime`, and `string`.

Functions ignore null values, such as those resulting from integer division by zero or empty data fields.

To add a `WHERE` clause to a query, use the query box in the data tab of the Controller Analytics search UI. When you perform the search, the visualization tab under the table widget shows the results of any metric functions.

Each section below describes one type of function.

Metric Functions

Metric functions:

- perform a calculation on a set of values for a field
- return a result such as a count or sum, or a set of results such as percentile
- can only be in the `SELECT` clause
- should always follow a field name if one exists—otherwise, not all visualizations may work

The `distinctcount()` function is accurate for up to 3000 unique return values. For over 3000 unique return values `distinctcount()` is approximate due to performance optimizations.

<table>
<thead>
<tr>
<th>Function</th>
<th>Returns</th>
<th>Valid Field Type(s)</th>
<th>ADQL Query Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>avg(numeric_field_name)</code></td>
<td>Average value for a numeric field.</td>
<td><code>integer</code>, <code>floating point</code>, <code>datetime</code></td>
<td><code>SELECT avg(bdata.estimatedtime) FROM browser_records WHERE appkey = 'your_appkey'</code></td>
</tr>
</tbody>
</table>
### count(field_name)
A count of events.
all

The form `count(*)` is also valid.

```
SELECT count(orderID) AS 'Sales' FROM logs WHERE method='GET'
```

The following query returns the top 10 IP addresses sending HTTP requests to an application.

```
SELECT IP, count(*) FROM web_session_records WHERE appkey = "yourApp" LIMIT 10
```

### distinctcount(field_name)
A count of the number of unique values recorded for a field.
all

```
SELECT distinctcount(field_name) FROM logs
```

### max(numeric_field_name)
The maximum recorded value of a numeric field.
integer, floating point, datetime

```
SELECT max(numeric_field_name) FROM transactions
```

```
SELECT nodeName, max(eventtimestamp) FROM logs
```

### min(numeric_field_name)
The minimum recorded value of a numeric field.
integer, floating point, datetime

```
SELECT min(numeric_field_name) FROM transactions
```

### stdev(numeric_field_name)
The population standard deviation of a numeric field.
integer, floating point

```
SELECT stdev(numeric_field_name) FROM transactions
```

### sum(numeric_field_name)
The sum of the values of a numeric field.
integer, floating point, datetime

```
SELECT sum(numeric_field_name) FROM transactions
```

### percentile(numeric_field_name, percent)
The specified percentile values between 0 and 100.
integer, floating point, datetime

```
SELECT percentile(responseTime, 50, 75, 90, 95) FROM transactions
```

### stats(numeric_field_name)
Statistics for a field: count, min, max, average, and sum.
integer, floating point, datetime

```
SELECT stats(numeric_fieldName) FROM transactions WHERE application='yourApp'
```

### totalResultCount()
The total count of events.
Does not accept a field.

```
SELECT transactionName, count ( "" ) / totalResultCount() FROM transactions WHERE application = 'ECommerce' AND userExperience = 'ERR OR'
```

## Bucketing Functions
Bucketing functions operate on a field, and group data into buckets. Often, the data in the buckets are further aggregated by another function. For example, this query creates buckets that group an app's transactions by the hour, and then count how many transactions are in each bucket (hour):

```
SELECT series(eventTimestamp, '1h'), count(*) FROM transactions WHERE application='yourAppName'
```

To get a larger data set in your results, use the time range selector in the upper right corner of the Controller Analytics search UI.

Dates are different from datetime values. Units from weeks down to milliseconds are supported, for example:

- '1w' for one week
- '10d' for ten days
- '2h' for two hours
- '5m' for five minutes
- '36s' for thirty-six seconds
- '800ms' for eight hundred milliseconds

When you use dates to define buckets, think about how many buckets your definition produces. The number of buckets for `series()` cannot exceed 2000, and much smaller numbers of buckets are the norm.

<table>
<thead>
<tr>
<th>Function</th>
<th>Returns</th>
<th>Valid Field Type(s)</th>
<th>ADQL Query Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>range(field_name, n</td>
<td>n1, n2), n3</td>
<td>n4, n5), ...)</td>
<td>One or more buckets.</td>
</tr>
<tr>
<td>series(field_name, interval_size)</td>
<td>A series of buckets based on the provided interval.</td>
<td>integer, floating point, dates</td>
<td>SELECT range(segments.transactionTime, 0, 10, 20) FROM transactions</td>
</tr>
<tr>
<td>series[field_name, interval_size, lower_extended_bound, upper_extended_bound, [offset],[rangeStyleBucketKeys],[strictEndpoints]]</td>
<td></td>
<td>Use a single value x or pairs of values (x, y). See below for explanation of how to define buckets.</td>
<td>Results in the following buckets: [-Inf, 0], [0, 10], [10, 20], [20, +Inf]</td>
</tr>
</tbody>
</table>

Defining buckets for `range()`

Every bucket is defined as bounded by two single values, where the bucket includes the first value and excludes the second.

Pairs of values (x, y) define an explicit bucket from x, inclusive, to y, exclusive.

A single value x implicitly defines a bucket according to rules that depend on where the value is placed among the arguments to `range()`:

- Rules for using a single value to define a bucket:
  a. Immediately after the field (that is, as the second argument): The bucket starts at negative infinity.
  b. After another single value: The bucket starts at the previous argument, inclusive, and extends to x, exclusive.
  c. After a pair of values, and before another single value: The bucket starts at x, inclusive, and extends to the
following argument, exclusive.
d. At the end of the parameter list: The bucket ends at positive infinity. This is the one exception to the assumption that \( x \) is the second of two values.
e. Between two pairs of values: No bucket can be defined, and the value is ignored.

**Range query example**

Here is a range query that exercises most of the rules above:

```sql
SELECT range(segments.transactionTime, 0, (10, 20), 30, (40, 50), 60, 70) FROM transactions
```

The query produces the following buckets:

- \([-\infty, 0]\) because rule a applies to 0
- \([10, 20]\) because \((10, 20)\) defines that bucket explicitly
- no bucket for 30 because rule e applies so 30 is ignored
- \([40, 50]\) because \((40, 50)\) defines that bucket explicitly
- \([60, 70]\) because rule c applies to 60
- \([70, +\infty]\) because rule d applies to 70

**Optional keyword arguments for series()**: `offset, rangeStyleBucketKeys, and strictEndpoints`

Series supports the following optional keyword arguments.

**offset**

Changes the start value of each series from 0 by the specified positive (+) or negative offset (-) duration, such as 1h for an hour, or 1d for a day. The values `alignStart` and `alignEnd` align buckets to the start and end of the extended bounds arguments respectively. The function must have extended bounds specified to use these special values. Defaults to 0. Accepts both integer and string values.

Examples:

```sql
SELECT series(responseTime, 1, 1, 10, offset="alignStart"), count(*) FROM transactions
```

```sql
SELECT series(responseTime, 100, 1, 1000, offset=1), count(*) FROM transactions
```

**rangeStyleBucketKeys**

Converts the start and end values in a numeric series to "bucketStart - bucketEnd" format as a string type. For example, "0-10", "10-20", and so on. This output this argument returns is similar to what returned by the range function. The series function must be acting on a numeric type to use this keyword argument. Defaults to false. Accepts a boolean only.

Example:

```sql
SELECT series(responseTime, 100, 1, 1000, rangeStyleBucketKeys=true), count(*) FROM transactions
```

**strictEndpoints**

Adds a filter on the query starting at the minimum extended bound and extending to the maximum extended bound. This ensures that no buckets outside of the extended bounds are present in the results. Defaults to false. Accepts a boolean only. Extended bound forces the histogram aggregation to start building intervals on a specific minimum value and expands the interval up to a given maximum value.

Example:

```sql
SELECT series(responseTime, 2, 1, 5, strictEndpoints=true, rangeStyleBucketKeys=true), count(*) FROM transactions
```

**Selecting the Top n Results for Time Series Queries**

Top n results is available for SaaS deployments only.

The series function can compute the global top \( n \) over an entire period. To set a top \( n \) limit, add a `LIMIT` clause to the end of your query:

```sql
SELECT series(eventTimestamp, "1m"), transactionName, count(*) FROM transactions LIMIT 1, 12
```
This example limits the query to the top 12 results of transactionName. The value 1 is applied to the `series` function, but because `series` is a bucketing function, it does not actually use the limit. You must include a value for the `series` function.

The global top n limit does not apply when queries:

- Contain a selection before the `series` function
  
  ```adql
  SELECT application, series(eventTimestamp, '1m'), transactionName, ...
  ```

- Have additional bucketing functions
  
  ```adql
  SELECT series(eventTimestamp, '1m'), range(responseTime, 0, 60, 120), transactionName, ...
  ```

- Include a `HAVING` clause
  
  ```adql
  SELECT series(eventTimestamp, '1m'), transactionName, avg(responseTime), ... HAVING a >15
  ```

String Functions

ADQL string manipulation functions:

- resemble their SQL counterparts
- are case-sensitive
- return null if any argument is null
- use 1-based, not 0-based, indexing
  - For example, the `indexOf()` function gives the index of the first character of a string as 1, not 0.
  - As a result, 0 is not a valid value for any integer argument

String functions can transform any event data that resides in the Analytics Events Service. One important use case is doing ad hoc data transformation when the structure, serialization, or format in which data has been collected needs to be changed.

Prior to AppDynamics 4.5.2, fixing incorrectly-collected data required adjusting collection settings.

You can use string functions to:

- hone in on a `substring()` by working with string `length()` and/or the `indexOf()` characters in a string
  - for example, you can extract the domain name from a URL
- combine multiple strings using the `concat()` operator
  - for example, you can combine values from multiple fields into a single value
- `trim()` whitespace from the beginnings and ends of fields
- pair the output of the `substring()` function, item by item, with the output of another function
  - this kind of implicit aggregation is the ADQL equivalent of GROUP BY in SQL
- see the Implicit Aggregation Example below

<table>
<thead>
<tr>
<th>Function</th>
<th>Returns</th>
<th>Valid Field Type(s)</th>
<th>ADQL Query Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>concat(inputString1, inputString2, ..., inputStringN)</code></td>
<td>A single string which combines the input strings.</td>
<td>string</td>
<td>Function:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>concat('foo', 'bar')</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Specify two or more strings.</td>
<td>Result:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>'foobar'</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
<td>Example</td>
<td>Result</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------</td>
<td>------------</td>
</tr>
</tbody>
</table>
| `indexOf(inputString, substring)` | 1-based start index of substring in string.  
If `substring` is not found, returns 0.  
If `occurrence` is supplied as n, returns index of nth occurrence of substring.  
If `occurrence` is supplied as -n, returns index of nth occurrence of substring counting backwards from the end of `inputString`. | `indexOf('www.wikipedia.org', '.', 1)` | 4          |
| `indexOf(inputString, substring, occurrence)` | string for inputString, substring  
positive or negative int for occurrence | `indexOf('www.wikipedia.org', '.', -1)` | 14         |
| `length(inputString)`             | The number of characters including whitespace of the input string.          | `length(' foo bar ')`                                                    | 'foo bar'  |
| `substring(inputString, startIndex, numChars)` | The substring of `inputString` starting at (1-based) `startIndex`.  
If `startIndex` is negative, starts that number of characters from the end of the string.  
If `numChars` is supplied, returns substring of that length; if `numChars` is past end of string, returns only up to end of string.  
If `startIndex` or `numChars` is invalid, returns an empty string. | `substring('https://example.com/home.htm', 9, 3)` | 'example.com' |
| `trim(inputString)`               | A copy of the input string with leading and trailing whitespace removed.    | `trim(' foo bar ')`                                                      | 'foo bar'  |

**Implicit aggregation example**

Consider the following data:

<table>
<thead>
<tr>
<th>Application ID</th>
<th>responseTime (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SJC-001</td>
<td>500</td>
</tr>
<tr>
<td>SJC-002</td>
<td>600</td>
</tr>
<tr>
<td>SJC-003</td>
<td>700</td>
</tr>
</tbody>
</table>
The following ADQL query pairs each item from the output of `substring()` with a corresponding item from the output of `avg()`:

```sql
SELECT substring(application, 1, 3), avg(responseTime) FROM transactions
```

The results are:

- SJC, 600
- LAX, 300

Notice how ADQL uses `substring()` to govern the groupings to which the second function, `avg()`, is applied.

### Data Type Conversion Functions

Data type conversion functions are only available for SaaS deployments.

Data type conversion functions can be used in the SELECT or WHERE clauses. Below are the available data type conversion functions. For `formatString` requirements, see Joda's `DateTimeFormat` documentation in the Classes section.

<table>
<thead>
<tr>
<th>Function</th>
<th>Returns</th>
<th>ADQL Query Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>toDate(string value, [string formatString])</code></td>
<td>A string value formatted as a datetime.</td>
<td><code>SELECT toDate(field_name, &quot;MM/dd/yyyy&quot;) FROM transactions</code></td>
</tr>
<tr>
<td></td>
<td>Returns</td>
<td><code>2019-01-15T00:00:00.000Z</code> for a field with string value &quot;01/15/2019&quot;</td>
</tr>
<tr>
<td><code>toDate(int value)</code></td>
<td>An integer value formatted as a datetime.</td>
<td><code>SELECT toDate(12341234) FROM transactions</code></td>
</tr>
<tr>
<td><code>toString(datetime value, [string formatString])</code></td>
<td>A datetime value formatted as a string.</td>
<td><code>SELECT toString(field_name, &quot;MM dd, yyyy&quot;) FROM transactions</code></td>
</tr>
<tr>
<td></td>
<td>Returns</td>
<td>&quot;Jan 15, 2019&quot; for a field with datetime value 2019-01-15T00:00:00.000Z</td>
</tr>
<tr>
<td><code>toString(float/int/bool value)</code></td>
<td>A value formatted as a string.</td>
<td><code>SELECT toString(123.123) FROM transactions</code></td>
</tr>
<tr>
<td></td>
<td>Returns</td>
<td>&quot;123.123&quot;</td>
</tr>
</tbody>
</table>
### Functions That Modify Other Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Returns</th>
<th>Valid Field Type(s)</th>
<th>ADQL Query Examples</th>
</tr>
</thead>
</table>
| toInt(datetime/float/string/bool value) | An integer value, formatted as follows:  
- datetime- converts to milliseconds,  
- float- truncates  
- string- parses to floats and truncates to an integer  
- bool- converts to 0/1 | SELECT toInt(123.4) FROM transactions  
Returns  
123 |  
| toFloat(int/string value) | An integer or string value formatted as a float. | SELECT toFloat(123123) FROM transactions  
Returns  
123123.0 |  
| round(float value, int decimalPlaces) | A value rounded off to decimalPlaces. decimalPlaces must be a non-negative integer. | SELECT round(123.123123, 3) FROM transactions  
Returns  
123.123 |  
| ifNull(object value, object replacementValue) | A value if the value is non-null. If the value is null, returns replacementValue. | SELECT ifNull(field_name, 10) FROM transactions  
Returns  
10 if the value of field_name is null  
Returns  
Value of field_name if the value is not null |  

For data type conversion functions (except ifNull), if the value of field_name returns null, the converted data type will also return null.

Using mathematical and `now()` functions on datetimes

You can perform addition and subtraction mathematical functions on datetime and long values, for example:

```sql
SELECT toDate(field_name, "MMM, dd, yyyy") - toDate(field_name, "MMM, dd, yyyy") FROM transactions
```

The function will return a datetime value. Multiplication, division, and modulus mathematical functions are not supported. Addition and subtraction functions cannot be combined with floats.

You can use the `now()` function to add or subtract the server's current time from a value, for example:

```sql
SELECT toInt(now() - field_name) FROM transactions
```

The above function determines how long ago `field_name` was created and converts the value to an integer.
<table>
<thead>
<tr>
<th>filter(metric_function (field expression), ...) [WHERE] condition_expression)</th>
<th>Aggregations for specific subsets of data, computed by applying a filter to the input of a single metric function.</th>
<th>The metric function must be one of the following that returns a single value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>filter(accumulate(field_expression), ...) [WHERE] condition_expression)</td>
<td>The second argument has the same syntax as a WHERE clause and optionally begins with the WHERE keyword, for example:</td>
<td><code>SELECT filter(avg(responseTime), WHERE responseTime &gt; 1) FROM transactions WHERE application = &quot;Travel&quot;</code></td>
</tr>
<tr>
<td>filter(accumulate(field_expression), ...) [WHERE] condition_expression)</td>
<td>filter(accumulate(field_expression), ...) [WHERE] condition_expression)</td>
<td><code>SELECT 100.0 * filter(count(*), field_name = &quot;value&quot;) / count(*) as &quot;%&quot;, FROM transactions WHERE application = &quot;yourApp&quot; AND transactionName = &quot;yourValue&quot;</code></td>
</tr>
</tbody>
</table>

### Combining Different Types of Functions

Here are some special cases to consider when combining functions of different types:

- The maximum accuracy of `distinctcount()` is reduced if it follows a bucketing function. Nested `distinctcount()` queries trade accuracy for memory savings, because the previous function establishes buckets, and `distinctcount()` creates sets of buckets within each of the previous function's buckets, and Elasticsearch automatically makes that tradeoff when that happens.
- For example, the first of the two example queries below returns less accurate results than the second:
  - `SELECT transactionName, distinctcount(userId) FROM transactions`  
  - `SELECT distinctcount(userId) FROM transactions`
- The return value of `stdev(numeric_field_name)` cannot be used to sort or filter results.
Analyzed Fields

Overview

Selected fields in the analytics data are "analyzed". Analysis consists of tokenizing a block of text into individual terms suitable for use in an inverted index and normalizing these terms into a standard form to improve their searchability. The ADQL Comparison Operators behave differently for analyzed and non-analyzed fields. To construct queries on analyzed fields, you need to understand some concepts about how the tokens are built.

Uppercase letters in analyzed fields are all converted to lowercase to build tokens.

Delimiters and other factors (such as CamelCase terms) affect how strings are tokenized. For a string such as "myname@company.com", "@" is a delimiter, therefore myname and company.com are two separate tokens. Additionally, company.com is two separate tokens. Note that all non-alphanumeric characters are delimiters. A term that uses CamelCase, such as VicePresident, is tokenized into separate tokens based on recognition of the CamelCase nature of the term resulting in the tokens: Vice and President as well as VicePresident.

For an example string such as: <VicePresident:SalesAndMarketing> - EMEA Australia USA 94107, the tokens generated include the following:

- vicepresident
- vice
- president
- salesandmarketing
- sales
- and
- marketing
- emeaaustraliausa94107
- emeaaustraliausa
- emea
- australia
- usa
- 94107

Analytics Analyzed Fields

The analyzed fields in analytics events are the following:

- Logs: Message
- Transactions: Errors and Error Detail
- Mobile: stacktrace

Queries on Analyzed Fields

Full-text search is supported on analyzed fields, including the message field for logs, using the LIKE operator. See Comparison Operators for details on using LIKE.

On analyzed fields, the REGEXP operator matches exactly only the analyzed and processed tokens, so you cannot query across the complete message.

Consider this log message:
In this log message, myname and company.com are two separate tokens because "@" is a delimiter. To search a log message like this for results based on the email address requires searching across tokens.

A query such as the following using REGEXP, will fail because myname and company.com are two separate tokens.

```
SELECT FROM logs WHERE appName = 'yourAppName' AND sourceType = 'yourLogFile' AND message REGEXP 'myname@company.*'
```

The LIKE operator is not affected by delimiters, so an alternative query using LIKE operator is a better choice.

```
SELECT FROM logs WHERE appName = 'yourAppName' AND sourceType = 'yourLogFile' AND message LIKE 'myname@company'
```

You can also use wildcards in the query because they work across tokens.

```
SELECT FROM logs WHERE appName = 'yourAppName' AND sourceType = 'yourLogFile' AND message LIKE 'myname@company*'
```

**Example Queries for Analyzed Fields**

Searching the following analytics log events:

```
Jun 7 11:27:45 appd sshd[1032]: Illegal user test from 110.49.183.11
Jun 7 11:27:46 appd sshd[1032]: Failed password for illegal user test from 110.49.183.11 port 9218 ssh2
Jun 7 11:27:46 appd sshd[1032]: error: Could not get shadow information for NOUSER
```
Note the results from the following queries:

<table>
<thead>
<tr>
<th>Query</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>SELECT * FROM logs WHERE sourceType='yourLog File' AND message REGEXP 'illegal.+user'</code></td>
<td>Does not match any log events in the sample because the query string spans across multiple tokens. Use LIKE for an instance like this.</td>
</tr>
<tr>
<td><code>SELECT * FROM logs WHERE sourceType='yourLog File' AND message REGEXP 'illegal.*'</code></td>
<td>_matches the first two log events</td>
</tr>
<tr>
<td><code>SELECT * FROM logs WHERE sourceType='yourLog File' AND message REGEXP 'Failed'</code></td>
<td>Does not match any log events in the sample because the token has only lowercase ‘failed’</td>
</tr>
</tbody>
</table>

To search for string : javaIOException

<table>
<thead>
<tr>
<th>Query</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>SELECT * FROM transactions WHERE application = 'yourApp' AND segments.errorList.errorCode REGEXP [a-z][a-z][a-z][a-z][a-z][a-z][a-z][a-z][a-z][a-z]</code></td>
<td></td>
</tr>
</tbody>
</table>
Math Expressions

ADQL Reference:

Overview

Math expressions are field expressions that use arithmetic operators. These expressions can be used in ADQL queries in both the SELECT and WHERE clauses.

field_expression BINARY_OPERATORS field_expression

Arithmetic Operators

Math operators are valid on integer and float data fields, constants (for example, 3.14), and on functions that return a single integer or float value, with the exception of % (modulo). The operators +, / and % have higher precedence than + and -. Parentheses can be used to control the order of operations, a unary minus (-) is also supported.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>asterisk</td>
<td>Multiplication</td>
</tr>
<tr>
<td>/</td>
<td>slash</td>
<td>Division of two integer types performed as integer division. Integer division by zero returns null.</td>
</tr>
<tr>
<td>%</td>
<td>modulo</td>
<td>Calculates the reminder after the division of two integers. Valid only for integers.</td>
</tr>
<tr>
<td>+</td>
<td>plus</td>
<td>Addition. Requires a space before a numeric literal.</td>
</tr>
<tr>
<td>-</td>
<td>minus</td>
<td>Subtraction. Requires a space before a numeric literal. Can also be used as unary_operator.</td>
</tr>
</tbody>
</table>

Math Expression Usage

Math expressions can be used in several ways.

<table>
<thead>
<tr>
<th>Query</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT regionId % 4, count(*) FROM transactions WHERE application = &quot;yourApp&quot;</td>
<td>Math expression is used as a GROUP BY term in an implicit group by query</td>
</tr>
<tr>
<td>SELECT field_name1, (field_name2 + field_name3) * 3.14 FROM transactions WHERE application = &quot;yourApp&quot;</td>
<td>Math expression is used as a composite field.</td>
</tr>
<tr>
<td>SELECT sum(field_name1) / count(field_name2) FROM transactions WHERE application = &quot;yourApp&quot;</td>
<td>Math expression used on an aggregation function.</td>
</tr>
<tr>
<td>SELECT filter(sum(field_name1), WHERE field2 IS NOT NULL) / count(field_name3) FROM ...</td>
<td></td>
</tr>
<tr>
<td>SELECT avg(field1 + field2) FROM event_type WHERE ...</td>
<td>Math expression used within an aggregation function.</td>
</tr>
<tr>
<td>SELECT series(field1 + field2, 1) FROM event_type WHERE ...</td>
<td></td>
</tr>
<tr>
<td>SELECT * FROM transactions WHERE (responseTime * cartValue) &gt; 100</td>
<td>Math expression used in a WHERE clause with comparison operators.</td>
</tr>
<tr>
<td>SELECT * FROM transactions WHERE (regionId % 3) IN (1, 2)</td>
<td></td>
</tr>
<tr>
<td>SELECT * FROM transactions WHERE (responseTime * cartValue) BETWEEN (100, 200)</td>
<td></td>
</tr>
<tr>
<td>SELECT segments.userData.itemTitle, (segments.userData.itemPrice / 100) AS Price FROM transactions WHERE application = 'ECommerce 2.0' AND transactionName = '/product_details.xhtml' AND Price &gt; 20</td>
<td>Convert a value in cents to dollars. This query converts segments.userData.itemPrice from cents to dollars and shows results where the price is greater than $20.</td>
</tr>
</tbody>
</table>
### Usage Notes

If any field or intermediate expression result for an expression is null, the entire expression will be null.

Math expressions in the **WHERE** clause are evaluated per record, not in aggregate.

Fields and metric functions cannot be combined in the same expression, for example, a query for the following type is **not** valid:

```sql
SELECT responseTime + sum(cartValue) FROM ...
```

Mathematical operations that operate on every event (as opposed to on aggregation functions) will reduce query performance. Avoid statements such as:

```sql
WHERE (responseTime - 200) > 800 and use a construct such as WHERE responseTime > 1000 instead.
```

```sql
SELECT transactionName, (responseTime / 1000) AS responseSeconds, 
userExperience, requestGUID FROM transactions WHERE application = 
"ECommerce 2.0" AND responseSeconds > 5
```

Convert a value such as average response time to seconds instead of milliseconds and view each transaction that is greater than X seconds.
Comparison operators, such as =, !=, <, >, LIKE, and IN, can be used in condition_expressions of the WHERE clause in the ADQL query statement.

This page lists the comparison operators that can be used in the condition_expression syntax.

### Operators

Comparisons on strings are case-insensitive.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Supported Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>Equals: True if the value of the specified field_name equals the specified value in the expression. String comparisons using the equals operator are case-sensitive.</td>
<td>non-analyzed fields</td>
</tr>
<tr>
<td>!=</td>
<td>Not equals: True if the value in the specified field_name does not equal the specified value.</td>
<td>non-analyzed fields</td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater than: True if the value in the specified field_name is greater than the specified value.</td>
<td>non-analyzed fields of type int, float and date</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Greater than or equal: True if the value in the specified field_name is greater than or equal to the specified value.</td>
<td>non-analyzed fields of type int, float and date</td>
</tr>
<tr>
<td>&lt;</td>
<td>Less than: True if the value in the specified field_name is less than the specified value.</td>
<td>non-analyzed fields of type int, float and date</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Less than or equals: True if the value in the specified field_name is less than or equal to the specified value.</td>
<td>non-analyzed fields of type int, float and date</td>
</tr>
<tr>
<td>IN</td>
<td>Finds records where the value of a field equals any one of the specified values in a list of string or numeric values. The values for IN must be in parentheses. String values must be surrounded by single quotes. Wild card usage within the list of values provided to the IN operator is supported. For example: <code>SQL SELECT transactionName, percentile(segments.transactionTime, 90) FROM transactions WHERE application = 'prd-analytics' AND transactionName IN ('api_v2./v2/events/*/search','api_v1./v1/events/*/search') </code></td>
<td>non-analyzed fields only. Invalid for analyzed fields.</td>
</tr>
<tr>
<td>LIKE</td>
<td>The LIKE operator performs a full-text search that analyzes the string value and field value. If the string value contains multiple terms, separated by spaces, it is processed as an AND condition of the terms. Use LIKE to search values in analyzed fields. The LIKE operator supports the use of wild cards. LIKE is not supported for queries on mixed case strings. For example, querying against a log message that contains &quot;pEer&quot; will not match the search string 'peer', because &quot;pEer&quot; is tokenized by the search analyzer into two separate tokens - &quot;p&quot; and &quot;eer&quot;).</td>
<td>for full-text search of analyzed fields</td>
</tr>
</tbody>
</table>
BETWEEN  Finds values of a field based on an inclusive or exclusive value range. For an inclusive range search, use square brackets [ ]. For an exclusive value range, use parentheses ( ).  Supported for data types int, float, and datetime (New in 4.4.1).

IS NULL  Finds records where a field does not contain a value.  non-analyzed fields and analyzed fields

IS NOT NULL  Finds records where a field contains a value.  analyzed and non-analyzed fields

NOT BETWEEN  Finds records where a field does not contain a value in an inclusive or exclusive value range.  The reverse of BETWEEN.  non-analyzed fields

NOT LIKE  The reverse of LIKE.  analyzed fields

NOT IN  Finds records where the value of a field does not match any value in a list of string or numeric values. The values for NOT IN must be in parentheses, and string values must be surrounded by single quotes.  There is also a logical operator NOT, which is unrelated to this comparison operator.

There is also a logical operator NOT, which is unrelated to this comparison operator.

The wildcard operators. Use of wild cards is supported on queries of string fields. The asterisk, "*", character matches zero or more characters and the question mark, "?", character matches a single character.

The wildcard operator works similarly when used with the equals "=" or LIKE operator.

The wildcard operator is case sensitive for non-analyzed fields.

REGEXP  The REGEXP operator applies a regular expression pattern match of a string to the pattern passed as an argument. See REGEXP Operator for details of usage. This is an advanced feature and requires knowledge of regular expression patterns.  non-analyzed fields

For analyzed fields:

- Use lowercase input strings.
- Do not use to search across tokens

Examples

<table>
<thead>
<tr>
<th>Query</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT * FROM mobile_snapshots WHERE appkey='your_appkey' AND numeric_field_name IN (1419, 613, 748)</td>
<td>Returns all fields for mobile snapshots for the specified appkey where the value of the specified numeric field matches one of the values in parentheses.</td>
</tr>
<tr>
<td>SELECT count(*) FROM browser_records WHERE appkey='your_appkey' AND field_name IN ('string1','string2', 'string3')</td>
<td>Returns a count of browser records for the specified appkey where the value of the specified field matches one of the specified strings.</td>
</tr>
<tr>
<td>SELECT ip FROM browser_records WHERE ip LIKE &quot;10.134.**&quot;</td>
<td>By filtering on IP address range using wild card operator, finds records in the specified IP address range.</td>
</tr>
<tr>
<td>SELECT * FROM logs WHERE message LIKE 'DEBUG Adding operation'</td>
<td>Matches the sample log line &quot;DEBUG com.appdynamics.operations.PublishTestService - Adding to bulk publish operation.&quot;</td>
</tr>
<tr>
<td>SELECT * FROM logs WHERE message LIKE 'error'</td>
<td>Finds all the log events containing the word 'Error' in any field</td>
</tr>
<tr>
<td>SELECT * FROM transactions WHERE application='AppName' AND cartTotal BETWEEN [10, 50]</td>
<td>Inclusive range: Returns all transactions with a cart total between 10 and 50, including 10 and 50.</td>
</tr>
<tr>
<td>SELECT * FROM transactions WHERE application='AppName' AND cartTotal BETWEEN (10, 50)</td>
<td>Exclusive range: Returns all transactions with cart total between 10 and 50, excluding 10 and 50.</td>
</tr>
</tbody>
</table>
Logical Operators

ADQL Reference:

**Overview**
Logical operators can be used in the WHERE clause in an ADQL query to make compound condition expressions. These operators are AND, OR, and NOT.

**Condition Expression Syntax**
(condition_expression LOGICAL_OPERATOR condition_expression)

**Examples**

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>(field_expression_1 AND field_expression_2)</td>
<td>Evaluates as true if both field_expressions are true.</td>
</tr>
<tr>
<td>(field_expression_1 OR field_expression_2)</td>
<td>Evaluates as true if either field_expression is true.</td>
</tr>
<tr>
<td>(field_expression_1 AND NOT field_expression_2)</td>
<td>Evaluates are true if field_expression_1 is true and field_expression_2 is false.</td>
</tr>
</tbody>
</table>
REGEXP Operator

ADQL Reference:

Overview

The regular expression operator, REGEXP, can be used in the WHERE clause to handle complex matching queries. This operator applies a regular expression pattern match of a string to the pattern passed as an argument. ADQL uses the Lucene regular expression engine to analyze the REGEXP expression. The search results for queries that use the REGEXP operator are dependent not only on regular expression syntax and rules but also on whether you are searching analyzed or non-analyzed fields. To use REGEXP on analyzed fields, be sure to read about data indexing and searching across tokens in this topic: Analyzed Fields.

The REGEXP operator can only match with the lowercased tokens in the analyzed fields. To search for an uppercase string in an analyzed field using a wild card or REGEXP, you need to input a lowercase string. For example, info, not INFO.

Allowed Characters

Any Unicode character may be used in the regular expression pattern, but certain characters are reserved and must be escaped. Any reserved character can be escaped with a backslash "\"" including a literal backslash character:"\".

The standard reserved characters are the following:

```
. ? + * | { } [ ] ( ) " \
```

For query performance reasons, before using any reserved character in a REGEXP pattern, you need to specify the first three characters of the string explicitly, such as specifying 123 before the brackets [0-9] in the following example:

```
SELECT * FROM logs WHERE sourceType='yourLogFile' AND id REGEXP '123 [0-9]'
```

A query such as the following is invalid.

```
SELECT * FROM logs WHERE sourceType='yourLogFile' AND id REGEXP '[0-9]'
```

Example Operations, Strings, and Associated Patterns

<table>
<thead>
<tr>
<th>Supported Operation</th>
<th>Description</th>
<th>String</th>
<th>Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>Match any character</td>
<td>A period &quot;.&quot; can be used to represent any character.</td>
<td>abcde</td>
<td>abc..</td>
</tr>
<tr>
<td>One or more</td>
<td>A plus sign &quot;+&quot; can be used to repeat the preceding shortest pattern one or more times.</td>
<td>aaabb</td>
<td>aaab+</td>
</tr>
<tr>
<td>Zero or more</td>
<td>An asterisk &quot;*&quot; can be used to match the preceding shortest pattern zero or more times.</td>
<td>aaabbbcc</td>
<td>aaab<em>b</em>c*</td>
</tr>
<tr>
<td>Zero or one</td>
<td>A question mark &quot;?&quot; makes the preceding shortest pattern optional. It matches zero or one times.</td>
<td>aaabbbcc</td>
<td>aaabbc</td>
</tr>
<tr>
<td>Min-to-max</td>
<td>Curly brackets &quot;{}&quot; can be used to specify a minimum and (optionally) a maximum number of times the preceding shortest pattern can repeat. Allowed forms: (5) # repeat exactly 5 times (2,5) # repeat at least twice and at most 5 times</td>
<td>aaabbbcc</td>
<td>aaab[3]b[2]</td>
</tr>
<tr>
<td>Grouping</td>
<td>Parentheses &quot;+()&quot; can be used to form sub-patterns.</td>
<td>abababab</td>
<td>abab(ab)*</td>
</tr>
<tr>
<td>Alternation</td>
<td>The pipe symbol &quot;</td>
<td>&quot; acts as an OR operator. The match will succeed if the pattern on either the left-hand side OR the right-hand side matches.</td>
<td>aaabb</td>
</tr>
</tbody>
</table>
ADQL Data

For details for the Analytics data event types described below, refer to the links to the right under ADQL Data.

Transaction Analytics refers to event data collected by the AppDynamics Java and .NET app agents. Data is collected for each instance of a business transaction. Each instance of a business transaction passing through a tier is an event. You can view the raw data for any transaction instance that occurred within the data retention time. This gives you the ability to analyze the business impact of every event and learn how your customers use your application.

Log Analytics provides analytics on data collected from log files.

Browser, Mobile and IoT Analytics provide analytics on data collected by the AppDynamics End User Monitoring (EUM) JavaScript Agent, Mobile, and IoT Monitoring SDKs.

Synthetic Sessions Analytics provides analytics on sessions data collected by AppDynamics Browser Synthetic Monitoring.

Analytics Custom Events can be added using the AppDynamics Analytics Events API.
Analytics Transaction Data

Each instance of a business transaction passing through a single tier is an event, so the data associated with a single execution of a distributed business transaction (for example, Checkout) is stored as a series of events in Analytics. The data consists of the default transaction data and additional data from data collectors configured for that transaction. This data is organized and stored per business transaction.

Default Transaction Data

Event Type: transactions

Key (event type identifier): application

The following table lists the data collected by default for each instance of the business transactions that are enabled for Analytics. The table also lists the Events Service internal name. When accessing the data using ADQL queries, you need to use the Events Service internal name in the query.

<table>
<thead>
<tr>
<th>UI Field Name</th>
<th>Description</th>
<th>Events Service Internal Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Application name</td>
<td>application</td>
</tr>
<tr>
<td>Business Transaction</td>
<td>Business transaction name</td>
<td>transactionName</td>
</tr>
<tr>
<td>Error</td>
<td>Error details (deprecated) This field is replaced by the more granular fields: Error Code, Error Detail, and Error Type.</td>
<td>deprecated</td>
</tr>
<tr>
<td>Error Code</td>
<td>The value of this field depends on the Error Type. For example, if Error Type is an HTTP_Error_code, then the value is a string representing the corresponding HTTP error code such as 404, 503 and so on.</td>
<td>segments.errorList.errorCode</td>
</tr>
<tr>
<td>Error Detail</td>
<td>This is what the &quot;Error&quot; field represented in 4.1. Primarily the error message</td>
<td>segments.errorList.errorDetail</td>
</tr>
<tr>
<td>Error Type</td>
<td>Classification of errors into buckets done internally by the Agent. For example, the error type could be HTTP_ERROR_CODE, LOGGER_MESSAGE, THROWABLE and so on.</td>
<td>segments.errorList.errorType</td>
</tr>
<tr>
<td>Node</td>
<td>Node name</td>
<td>segments.node</td>
</tr>
<tr>
<td>Tier</td>
<td>Tier name</td>
<td>segments.tier</td>
</tr>
<tr>
<td>requestGUID</td>
<td>GUID for this specific user request as assigned by AppDynamics</td>
<td>requestGUID</td>
</tr>
<tr>
<td>Response Time (ms)</td>
<td>Business transaction response time for this request in milliseconds</td>
<td>responseTime</td>
</tr>
<tr>
<td>Timestamp</td>
<td>Time the event occurred in the application</td>
<td>eventTimestamp</td>
</tr>
<tr>
<td>User Experience</td>
<td>Indicates if the transaction was marked as Normal, Slow, Very Slow, Stall, or Error</td>
<td>userExperience</td>
</tr>
</tbody>
</table>
Exit calls | Details of database and remote service calls. View Details flow map. Note that for partial snapshots, exit call information captured in call graphs can differ between the Analytics and APM UI. Each UI presents an appropriate level of contextual visibility while preserving low overhead. An APM snapshot, for example, may capture only the exit calls that are most pertinent to transaction performance. | Not Applicable

Custom HTTP Request Data

When HTTP data collectors are configured, the following information can be collected:

<table>
<thead>
<tr>
<th>Field</th>
<th>Events Service Database Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>cookies</td>
<td>segments.httpData.cookies</td>
</tr>
<tr>
<td>headers</td>
<td>segments.httpData.headers</td>
</tr>
<tr>
<td>parameters</td>
<td>segments.httpData.parameters</td>
</tr>
<tr>
<td>principal</td>
<td>segments.httpData.principal</td>
</tr>
<tr>
<td>session ID</td>
<td>segments.httpData.sessionId</td>
</tr>
<tr>
<td>session Objects</td>
<td>segments.httpData.sessionObjects</td>
</tr>
<tr>
<td>URL</td>
<td>segments.httpData.url</td>
</tr>
<tr>
<td>URI path segments: segment0-n</td>
<td>segments.httpData.uriPathSegments</td>
</tr>
</tbody>
</table>

Analytics uses the configured display name as the field name for the HTTP parameters. In Controller versions prior to 4.3, Analytics displayed the data using the actual parameter name.

Custom Method Data

Custom data collected as specified in the method invocation data collector configuration. See Data Collectors for more details.

To specify a custom data field when using the Analytics query APIs, prefix the field name with segments.userData. For example, if your field is cartTotal, such as shown in the following screenshot, the database name is segments.userData.cartTotal.
Deprecated Fields

The two fields showing as Deprecated Fields are redundant with “Transaction Time” and “User Experience”. The “Transaction Time” and “User Experience” fields should be used preferentially if you have 4.2 Analytics Agents or higher deployed. The deprecated fields contain correct values but will not be supported in the future. Because the deprecated fields contain the right data from 4.1 agents, they are of value if you have 4.1 Analytics Agents reporting to a 4.2.8 or higher Controller and Events Service.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request Experience</td>
<td>Indicates if the transaction was marked as Normal, Slow, Very Slow, Stall, or Error</td>
</tr>
<tr>
<td>Transaction Time</td>
<td>Business transaction response time for this request in milliseconds.</td>
</tr>
</tbody>
</table>
Analytics Log Data

ADQL Reference:

Analytics data collected from log files depends on the source of the log file and the pattern that you specify for structuring the data in the log. Each log entry is an event in the Log Analytics event stream.

Log Analytics Data

Event Type: logs

Key (event type identifier): sourceType

The following fields are captured by default. You can configure and capture optional data, but the following fields are always present.

<table>
<thead>
<tr>
<th>UI Field Name</th>
<th>Description</th>
<th>Events Service Internal Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>pickupTimestamp</td>
<td>The timestamp when the agent picked up the event and sent it to the Analytics Agent</td>
<td>pickupTimestamp</td>
</tr>
<tr>
<td>Message</td>
<td>The message body of the log event</td>
<td>message</td>
</tr>
<tr>
<td>host</td>
<td>IP address or host name where the event was generated</td>
<td>host</td>
</tr>
<tr>
<td>source</td>
<td>Location of the logs, usually a path or directory such as /tomcat/logs</td>
<td>source</td>
</tr>
<tr>
<td>sourceType</td>
<td>The kind of log file, such as apache-httpserver-access-log</td>
<td>sourceType</td>
</tr>
<tr>
<td>Timestamp</td>
<td>Timestamp of the log event</td>
<td>eventTimestamp</td>
</tr>
</tbody>
</table>

Extracted Fields

Fields that were extracted using the Controller UI in previous versions appear in the Extracted Fields list. For details, see the section "Manage Extracted Fields" in Collect Log Analytics Data.

<table>
<thead>
<tr>
<th>Optional Fields</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nodeName (optional)</td>
<td>Name of the node where the log event occurred</td>
</tr>
<tr>
<td>tierName (optional)</td>
<td>Name of the tier where the log event occurred</td>
</tr>
<tr>
<td>appName (optional)</td>
<td>Application name where the log event occurred</td>
</tr>
</tbody>
</table>
Analytics Custom Events Data

Data collected for your analytics custom events depends on the schema you create and manage.

The timestamp is the only default field. All other fields are defined by your schema.

See the Analytics Events API documentation for more details on creating and querying custom events.
Analytics Browser Requests Data

ADQL Reference:

Browser Analytics provides details about each browser user request. The data collected includes information about pages, performance metrics, location of your users, browser and device data, errors, and any custom data that you configured in your Browser RUM configuration.

This page lists the field names available for browser request Analytics data, along with the Events Service internal name. When accessing the data using ADQL queries, you need to use the Events Service internal name in the query. For details on the meaning of each metric, see Browser RUM Metrics.

Event Type: browser_records

Key (event type identifier): appkey

<table>
<thead>
<tr>
<th>UI Field Name</th>
<th>Events Service Internal Name</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Page Info Fields</strong></td>
<td></td>
</tr>
<tr>
<td>Agent ID</td>
<td>agentid</td>
</tr>
<tr>
<td>appkey</td>
<td>appkey</td>
</tr>
<tr>
<td>Client GUID</td>
<td>cguid</td>
</tr>
<tr>
<td>IP address</td>
<td>ip</td>
</tr>
<tr>
<td>Page Experience</td>
<td>pageexperience</td>
</tr>
<tr>
<td>Page Name</td>
<td>pagename</td>
</tr>
<tr>
<td>Page Type</td>
<td>pagetype</td>
</tr>
<tr>
<td>Parent Page URL</td>
<td>pageparenturl - see note below</td>
</tr>
<tr>
<td>URL</td>
<td>pageurl</td>
</tr>
<tr>
<td><strong>Performance Time Fields</strong></td>
<td></td>
</tr>
<tr>
<td>Application Server Time</td>
<td>metrics.<code>Application Server Time (ms)</code></td>
</tr>
<tr>
<td>DOM Ready Time</td>
<td>metrics.<code>DOM Ready Time (ms)</code></td>
</tr>
<tr>
<td>Domain Lookup Time</td>
<td>metrics.<code>Domain Lookup Time (ms)</code></td>
</tr>
<tr>
<td>End User Response Time</td>
<td>metrics.<code>End User Response Time (ms)</code></td>
</tr>
<tr>
<td>Estimated BT Time</td>
<td>btdata.estimatedtime</td>
</tr>
<tr>
<td>First Byte Time</td>
<td>metrics.<code>First Byte Time (ms)</code></td>
</tr>
<tr>
<td>Server Connection Time</td>
<td>metrics.<code>Server Connection Time (ms)</code></td>
</tr>
</tbody>
</table>
### SSL Handshake Time

<table>
<thead>
<tr>
<th>Location Fields</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>City</td>
<td>geocity</td>
</tr>
<tr>
<td>Country</td>
<td>geocountry</td>
</tr>
<tr>
<td>Region</td>
<td>georegion</td>
</tr>
</tbody>
</table>

### Browser and Device Data Fields

<table>
<thead>
<tr>
<th>Browser and Device Data Fields</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Browser</td>
<td>browser</td>
</tr>
<tr>
<td>Browser Version</td>
<td>browserversion</td>
</tr>
<tr>
<td>Device</td>
<td>device</td>
</tr>
<tr>
<td>Device OS</td>
<td>deviceos</td>
</tr>
</tbody>
</table>

### Errors Field

<table>
<thead>
<tr>
<th>Errors Field</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AJAX Error</td>
<td>ajaxerror</td>
</tr>
<tr>
<td>Error Type</td>
<td>errortype</td>
</tr>
</tbody>
</table>

### Custom User Data Field

<table>
<thead>
<tr>
<th>Custom User Data Field</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>your_field_name</td>
<td>userdata.your_field_name</td>
</tr>
<tr>
<td></td>
<td>userdataBoolean.your_field_name</td>
</tr>
<tr>
<td></td>
<td>userdataDate.your_field_name</td>
</tr>
<tr>
<td></td>
<td>userdataLong.your_field_name</td>
</tr>
<tr>
<td></td>
<td>userdataDouble.your_field_name</td>
</tr>
</tbody>
</table>

Note: if your field name contains spaces or any reserved characters, the field name must be surrounded by backquotes. For example, if you define a custom field called “Item Purchased”, the Events Service field name is `userdata."Item Purchased"`.

The Parent Page URL field, `pageparenturl`, is only available after taking special action. On-premises customers need to run a special script on the EUM Server. SaaS customers need to file a ticket requesting AppDynamics to do this for them. See the Community Knowledge Base article: [https://community.appdynamics.com/t5/Knowledge-Base/Include-the-Page-Referrer-in-the-Browser-Requests-Analytics-Data/ta-p/28765](https://community.appdynamics.com/t5/Knowledge-Base/Include-the-Page-Referrer-in-the-Browser-Requests-Analytics-Data/ta-p/28765).
Analytics Browser Sessions Data

**ADQL Reference:**

This page lists the field names available for Analytics Browser Sessions data, along with the Events Service internal name. When accessing the data using ADQL queries, you need to use the Events Service internal name. The Browser Sessions data can be viewed using the Analytics Query API. This page lists the fields that are specific to Browser Sessions.

**Event Type:** web_session_records

**Key (event type identifier):** appkey

<table>
<thead>
<tr>
<th>UI Field Name</th>
<th>Events Service Internal Name</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Session Fields</strong></td>
<td></td>
</tr>
<tr>
<td>Browser</td>
<td>browser</td>
</tr>
<tr>
<td>Browser Version</td>
<td>browserversion</td>
</tr>
<tr>
<td>City</td>
<td>geocity</td>
</tr>
<tr>
<td>Closed</td>
<td>closed</td>
</tr>
<tr>
<td>Country</td>
<td>geocountry</td>
</tr>
<tr>
<td>Device</td>
<td>device</td>
</tr>
<tr>
<td>Device OS</td>
<td>deviceos</td>
</tr>
<tr>
<td>IP Address</td>
<td>ip</td>
</tr>
<tr>
<td>Page View Count</td>
<td>pageviews</td>
</tr>
<tr>
<td>Region</td>
<td>georegion</td>
</tr>
<tr>
<td>Session Duration (ms)</td>
<td>metrics.durationMS</td>
</tr>
<tr>
<td>Session Start Time</td>
<td>startTimeMS</td>
</tr>
<tr>
<td>User Experience</td>
<td>experience</td>
</tr>
<tr>
<td><strong>Page View Fields</strong></td>
<td></td>
</tr>
<tr>
<td>Client GUID</td>
<td>browserRecords.cguid</td>
</tr>
<tr>
<td>Page Error Type</td>
<td>browserRecords.errortype</td>
</tr>
<tr>
<td>Page Experience</td>
<td>browserRecords.pageexperience</td>
</tr>
<tr>
<td>RUM Page Name</td>
<td>browserRecords.pagename</td>
</tr>
</tbody>
</table>

**Custom User Data**
<table>
<thead>
<tr>
<th>your_field_name</th>
<th>userdata.your_field_name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>userdataBoolean.your_field_name</td>
</tr>
<tr>
<td></td>
<td>userdataDate.your_field_name</td>
</tr>
<tr>
<td></td>
<td>userdataLong.your_field_name</td>
</tr>
<tr>
<td></td>
<td>userdataDouble.your_field_name</td>
</tr>
</tbody>
</table>

Note: if your field name contains spaces or any reserved characters, the field name must be surrounded by backquotes. For example, if you define a custom field called "Item Purchased", the Events Service field name is `userdata."Item Purchased`.
Analytics Mobile Crash Reports Data

**ADQL Reference:**

This page lists the field names available for Analytics Mobile Crash Reports data, along with the Events Service internal name. When accessing the data using ADQL queries, you need to use the Events Service internal name in the query.

Event Type: mobile_crash_reports

Key (event type identifier): appkey

<table>
<thead>
<tr>
<th>UI Field Name</th>
<th>Events Service Internal Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>App Crash Time</td>
<td>appcrashtimestamp</td>
</tr>
<tr>
<td>App Key</td>
<td>appkey</td>
</tr>
<tr>
<td>Carrier</td>
<td>carrier</td>
</tr>
<tr>
<td>Connection Type</td>
<td>connectiontype</td>
</tr>
<tr>
<td></td>
<td>For example: &quot;wifi&quot;, &quot;4g&quot;, &quot;cell&quot;</td>
</tr>
<tr>
<td>Country</td>
<td>geocountry</td>
</tr>
<tr>
<td>Crash Group ID</td>
<td>groupid</td>
</tr>
<tr>
<td>Crash Id</td>
<td>crashid</td>
</tr>
<tr>
<td>Crashed Function</td>
<td>crashfunction</td>
</tr>
<tr>
<td></td>
<td>For example: &quot;.[MTFirstViewController segFault413]&quot;,</td>
</tr>
<tr>
<td></td>
<td>&quot;com.appd.examples.android.instrumentation.MainActivity.d42&quot;</td>
</tr>
<tr>
<td></td>
<td>This value depends on your application.</td>
</tr>
<tr>
<td>Device / Manufacturer</td>
<td>devicemanufacturer</td>
</tr>
<tr>
<td></td>
<td>For example: &quot;Samsung&quot;, &quot;Apple&quot;</td>
</tr>
<tr>
<td>Mobile App Name</td>
<td>mobileappname</td>
</tr>
<tr>
<td>Mobile App Version</td>
<td>mobileappversion</td>
</tr>
<tr>
<td>Model</td>
<td>devicemodel</td>
</tr>
<tr>
<td></td>
<td>For example: &quot;iPhone 5&quot; &quot;iPad 2 WIFI&quot;</td>
</tr>
<tr>
<td>OS Version</td>
<td>osversion</td>
</tr>
<tr>
<td></td>
<td>For example: &quot;iOS 6.0&quot;, &quot;Android 4.2&quot;</td>
</tr>
<tr>
<td>Platform</td>
<td>platform</td>
</tr>
<tr>
<td></td>
<td>For example: &quot;Android&quot;, &quot;iOS&quot;</td>
</tr>
<tr>
<td>Region</td>
<td>georegion</td>
</tr>
</tbody>
</table>

*User Data*
<table>
<thead>
<tr>
<th>your_field_name</th>
<th>userdata.your_field_name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>userdataBoolean.your_field_name</td>
</tr>
<tr>
<td></td>
<td>userdataDate.your_field_name</td>
</tr>
<tr>
<td></td>
<td>userdataLong.your_field_name</td>
</tr>
<tr>
<td></td>
<td>userdataDouble.your_field_name</td>
</tr>
</tbody>
</table>

Note: if your field name contains spaces or any reserved characters, the field name must be surrounded by backquotes. For example, if you define a custom field called "Item Purchased", the Events Service field name is `userdata."Item Purchased"`
Analytics Mobile Requests Data

**ADQL Reference:**

Mobile Analytics provides details about the performance of your mobile apps as experienced by your end users. The data collected includes information about the mobile app names and versions, network requests, performance times, locations, carrier and device data, errors, and any custom data that you configured in your Mobile RUM configuration.

This page lists the field names available for mobile requests Analytics data, along with the Events Service internal name. For details on the meaning of each metric, see Mobile RUM Metrics. When accessing the data using ADQL queries, you need to use the Events Service internal name in the query.

Event Type: mobile_snapshots

Key (event type identifier): appkey

<table>
<thead>
<tr>
<th>UI Field Name</th>
<th>Events Service Internal Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>agentid</td>
<td>agentid</td>
</tr>
<tr>
<td>agentversion</td>
<td>agentversion</td>
</tr>
<tr>
<td>appkey</td>
<td>appkey</td>
</tr>
<tr>
<td>cguid</td>
<td>cguid</td>
</tr>
<tr>
<td>Mobile App Name</td>
<td>mobileappname</td>
</tr>
<tr>
<td>Mobile App Version</td>
<td>mobileappversion</td>
</tr>
<tr>
<td>Platform</td>
<td>platform</td>
</tr>
</tbody>
</table>

**Network Request Fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Internal Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Background</td>
<td>happenedinbackground</td>
</tr>
<tr>
<td>Experience</td>
<td>networkrequestexperience</td>
</tr>
<tr>
<td>HTTP Status Code</td>
<td>httpstatuscode</td>
</tr>
<tr>
<td>Network Request Name</td>
<td>networkrequestname</td>
</tr>
<tr>
<td>Network Request Type</td>
<td>networkrequesttype</td>
</tr>
<tr>
<td>Request Content Length</td>
<td>requestContentLength</td>
</tr>
<tr>
<td>Response Content Length</td>
<td>responseContentLength</td>
</tr>
</tbody>
</table>

**Performance Time Fields**
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimated BT Time</td>
<td>btdata.estimatedtime</td>
</tr>
<tr>
<td>Network Request Time</td>
<td>networkrequesttime</td>
</tr>
</tbody>
</table>

**Location Fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Country</td>
<td>geocountry</td>
</tr>
<tr>
<td>Region</td>
<td>georegion</td>
</tr>
</tbody>
</table>

**Carrier and Device Data Fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carrier</td>
<td>carrier</td>
</tr>
<tr>
<td>Connection Type</td>
<td>connectiontype</td>
</tr>
<tr>
<td></td>
<td>For example: &quot;wifi&quot;, &quot;4g&quot;, &quot;cell&quot;</td>
</tr>
<tr>
<td>Device / Manufacturer</td>
<td>devicename</td>
</tr>
<tr>
<td></td>
<td>For example: &quot;Samsung&quot;, &quot;Apple&quot;</td>
</tr>
<tr>
<td>Model</td>
<td>devicemodel</td>
</tr>
<tr>
<td></td>
<td>For example: &quot;iPhone 5&quot; &quot;iPad 2 WIFI&quot;</td>
</tr>
<tr>
<td>OS Version</td>
<td>osversion</td>
</tr>
<tr>
<td></td>
<td>For example: &quot;iOS 6.0&quot;, &quot;Android 4.2&quot;</td>
</tr>
</tbody>
</table>

**Error Fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network Type</td>
<td>networkerror</td>
</tr>
</tbody>
</table>

**Custom User Data Fields**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>your_field_name</td>
<td>userdata.your_field_name</td>
</tr>
<tr>
<td></td>
<td>userdataBoolean.your_field_name</td>
</tr>
<tr>
<td></td>
<td>userdataDate.your_field_name</td>
</tr>
<tr>
<td></td>
<td>userdataLong.your_field_name</td>
</tr>
<tr>
<td></td>
<td>userdataDouble.your_field_name</td>
</tr>
</tbody>
</table>

Note: If your field name contains spaces or any reserved characters, the field name must be surrounded by backquotes. For example, if you define a custom field called "Item Purchased", the Events Service field name is `userdata.`'Item Purchased`
# Analytics Mobile Sessions Data

**ADQL Reference:**

This page lists the field names available for Analytics Mobile Sessions data, along with the Events Service internal name. When accessing the data using ADQL queries, you need to use the Events Service internal name in the query.

Event Type: mobile_session_records

Key (event type identifier): appkey

<table>
<thead>
<tr>
<th>UI Field Name</th>
<th>Events Service Internal Name</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mobile Session Fields</strong></td>
<td></td>
</tr>
<tr>
<td>Mobile App Name</td>
<td>mobileappname</td>
</tr>
<tr>
<td>example: &quot;com.appdynamics.eum.test.apps.E2E-PictureSharingApp&quot;</td>
<td></td>
</tr>
<tr>
<td>Carrier</td>
<td>carrier</td>
</tr>
<tr>
<td>For example: &quot;AT&amp;T&quot;, &quot;Verizon&quot;</td>
<td></td>
</tr>
<tr>
<td>Closed</td>
<td>closed</td>
</tr>
<tr>
<td>Connection Type</td>
<td>connectiontype</td>
</tr>
<tr>
<td>For example: &quot;cell&quot;, &quot;3g&quot;</td>
<td></td>
</tr>
<tr>
<td>Country</td>
<td>geocountry</td>
</tr>
<tr>
<td>Device / Manufacturer</td>
<td>devicename</td>
</tr>
<tr>
<td>Duration</td>
<td>metrics.durationMS</td>
</tr>
<tr>
<td>IP Address</td>
<td>ip</td>
</tr>
<tr>
<td>Mobile App Version</td>
<td>mobileappversion</td>
</tr>
<tr>
<td>Model</td>
<td>devicemodel</td>
</tr>
<tr>
<td>For example: &quot;iPad 2 WIFI&quot;, &quot;Kindle Fire&quot;</td>
<td></td>
</tr>
<tr>
<td>OS Version</td>
<td>osversion</td>
</tr>
<tr>
<td>For example: &quot;iOS 5.1&quot;, &quot;Android 4.2&quot;</td>
<td></td>
</tr>
<tr>
<td>Region</td>
<td>georegion</td>
</tr>
<tr>
<td>Session Start Time</td>
<td>startTimeMS</td>
</tr>
<tr>
<td>Session Status</td>
<td>closed</td>
</tr>
<tr>
<td>Note &quot;closed&quot; is a Boolean field.</td>
<td></td>
</tr>
<tr>
<td>True means the session is closed.</td>
<td></td>
</tr>
<tr>
<td>False means the session is still going on.</td>
<td></td>
</tr>
<tr>
<td><strong>Event Count Fields</strong></td>
<td></td>
</tr>
<tr>
<td>Metric Type</td>
<td>Metric Name</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>---------------------------------------</td>
</tr>
<tr>
<td>Breadcrumb Count</td>
<td>metrics.breadcrumbcount</td>
</tr>
<tr>
<td>Crash Count</td>
<td>metrics.crashcount</td>
</tr>
<tr>
<td>Custom Metric Count</td>
<td>metrics.custommetriccount</td>
</tr>
<tr>
<td>Info Point Count</td>
<td>metrics.infopointcount</td>
</tr>
<tr>
<td>Network Request Count</td>
<td>metrics.networkrequestcount</td>
</tr>
<tr>
<td>Timer Count</td>
<td>metrics.timerCount</td>
</tr>
<tr>
<td>UI Event Count</td>
<td>metrics.uiCount</td>
</tr>
<tr>
<td>User Data Count</td>
<td>metrics.userdatacount</td>
</tr>
</tbody>
</table>

**Crash Fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Metric Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>App Crash Time</td>
<td>crash.appcrashtimestamp</td>
</tr>
<tr>
<td>Crash Exception</td>
<td>crash.crashexception</td>
</tr>
<tr>
<td>Crash Group ID</td>
<td>crash.groupid</td>
</tr>
<tr>
<td>Crash Line Number</td>
<td>crash.crashlinenumber</td>
</tr>
<tr>
<td>Symbolicated or Deobfuscate</td>
<td>crash.symbolicateddeobfuscated</td>
</tr>
</tbody>
</table>

**User Data**

<table>
<thead>
<tr>
<th>Field</th>
<th>Metric Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>your_field_name</td>
<td>userdata.your_field_name</td>
</tr>
<tr>
<td></td>
<td>userdataBoolean.your_field_name</td>
</tr>
<tr>
<td></td>
<td>userdataDate.your_field_name</td>
</tr>
<tr>
<td></td>
<td>userdataLong.your_field_name</td>
</tr>
<tr>
<td></td>
<td>userdataDouble.your_field_name</td>
</tr>
</tbody>
</table>

*Note: if your field name contains spaces or any reserved characters, the field name must be surrounded by backquotes. For example, if you define a custom field called "Item Purchased", the Events Service field name is `userdata."Item Purchased"`*

**Breadcrumb Fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Metric Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breadcrumb Text</td>
<td>breadcrumb.bctext</td>
</tr>
</tbody>
</table>

**Network Request Fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Metric Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection Type</td>
<td>networkrequest.connectiontype</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>End Time</td>
<td>endTimeMS</td>
</tr>
<tr>
<td>Experience</td>
<td>networkrequest.networkrequestexperience</td>
</tr>
<tr>
<td>HTTP Status Code</td>
<td>networkrequest.httpstatuscode</td>
</tr>
<tr>
<td>IP Address</td>
<td>networkrequest.ip</td>
</tr>
<tr>
<td>Network Error</td>
<td>networkrequest.networkerror</td>
</tr>
<tr>
<td>Network Request Name</td>
<td>networkrequest.networkrequestname</td>
</tr>
<tr>
<td>Network Request Time</td>
<td>networkrequest.networkrequesttime</td>
</tr>
<tr>
<td><strong>Infopoint Fields</strong></td>
<td></td>
</tr>
<tr>
<td>Duration (ms)</td>
<td>infopoint.duration</td>
</tr>
<tr>
<td>infopname</td>
<td>infopoint.infopname</td>
</tr>
<tr>
<td>Is Dynamic</td>
<td>infopoint.isdynamic</td>
</tr>
<tr>
<td><strong>UI Event Fields</strong></td>
<td></td>
</tr>
<tr>
<td>Activity</td>
<td>ui.activity</td>
</tr>
<tr>
<td>Class</td>
<td>ui.uiclass</td>
</tr>
<tr>
<td></td>
<td>For example: &quot;UITableView&quot;</td>
</tr>
<tr>
<td>Label</td>
<td>ui.uilabel</td>
</tr>
<tr>
<td></td>
<td>For example: &quot;IOS TableCell&quot;</td>
</tr>
<tr>
<td>Root View</td>
<td>ui.rootview</td>
</tr>
<tr>
<td></td>
<td>For example: &quot;SettingsScreen&quot;</td>
</tr>
<tr>
<td>UI Event</td>
<td>ui.uievent</td>
</tr>
<tr>
<td></td>
<td>For example: &quot;Button Pressed&quot;, &quot;Root View Change&quot;, &quot;Table Cell Selected&quot;</td>
</tr>
</tbody>
</table>
### Analytics Mobile Non-Fatal Issues Data

**ADQL Reference:**

This page lists the field names available for Analytics Mobile Non-Fatal Issues data, along with the Events Service internal name. When accessing the data using ADQL queries, you need to use the Events Service internal name in the query.

Event Type: `mobile_non_fatal_issue_records`

<table>
<thead>
<tr>
<th>UI Field Name</th>
<th>Events Service Internal Name</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Agent Info Fields</strong></td>
<td></td>
</tr>
<tr>
<td>Agent ID</td>
<td><code>agentid</code></td>
</tr>
<tr>
<td>Agent Version</td>
<td><code>agentversion</code></td>
</tr>
<tr>
<td><strong>App Info Fields</strong></td>
<td></td>
</tr>
<tr>
<td>App Key</td>
<td><code>appkey</code></td>
</tr>
<tr>
<td>Mobile App Name</td>
<td><code>mobileappname</code></td>
</tr>
<tr>
<td></td>
<td>For example: “com.appdynamics.eum.test.apps.E2E-PictureSharingApp”</td>
</tr>
<tr>
<td>Mobile App Version</td>
<td><code>mobileappversion</code></td>
</tr>
<tr>
<td><strong>Device Info Fields</strong></td>
<td></td>
</tr>
<tr>
<td>Carrier</td>
<td><code>carrier</code></td>
</tr>
<tr>
<td></td>
<td>For example: “AT&amp;T”, “Verizon”</td>
</tr>
<tr>
<td>Connection Type</td>
<td><code>connectiontype</code></td>
</tr>
<tr>
<td></td>
<td>For example: “cell”, “3g”</td>
</tr>
<tr>
<td>Device Manufacturer</td>
<td><code>devicemanager</code></td>
</tr>
<tr>
<td></td>
<td>For example: “Samsung”</td>
</tr>
<tr>
<td>Device Model</td>
<td><code>devicemodel</code></td>
</tr>
<tr>
<td></td>
<td>For example: “Galaxy S5”</td>
</tr>
<tr>
<td>Device Name</td>
<td><code>devicename</code></td>
</tr>
<tr>
<td>IP Address</td>
<td><code>ip</code></td>
</tr>
<tr>
<td>jailbroken</td>
<td><code>jailbroken</code></td>
</tr>
<tr>
<td>OS Version</td>
<td><code>osversion</code></td>
</tr>
<tr>
<td></td>
<td>For example: “5.1”, “4.2”</td>
</tr>
<tr>
<td>Platform</td>
<td><code>platform</code></td>
</tr>
<tr>
<td></td>
<td>For example: “iOS”, “Android”</td>
</tr>
<tr>
<td><strong>Code Issues</strong></td>
<td></td>
</tr>
<tr>
<td>description</td>
<td><code>description</code></td>
</tr>
<tr>
<td></td>
<td>For example: “App Not Responding in Utils.formatData”</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>------------------------------------------</td>
</tr>
<tr>
<td>filename</td>
<td></td>
</tr>
<tr>
<td>groupid</td>
<td></td>
</tr>
<tr>
<td>Line Number</td>
<td></td>
</tr>
<tr>
<td>severity</td>
<td>For example: “Info”, “warning”</td>
</tr>
<tr>
<td>symbolicateddeobfuscated</td>
<td>For example: false</td>
</tr>
<tr>
<td>symbolname</td>
<td>For example: “MainActivity.onResume”</td>
</tr>
<tr>
<td>pickupTimestamp</td>
<td></td>
</tr>
<tr>
<td>Timestamp</td>
<td>For example: “03/15/18 5:32:33 PM”</td>
</tr>
</tbody>
</table>

**Custom User Data Fields**

- **your_field_name**
  - `userdata.your_field_name`
  - `userdataBoolean.your_field_name`
  - `userdataDate.your_field_name`
  - `userdataLong.your_field_name`
  - `userdataDouble.your_field_name`

  Note: if your field name contains spaces or any reserved characters, the field name must be surrounded by backquotes. For example, if you define a custom field called “Item Purchased”, the Events Service field name is `userdata.`Item Purchased`

**Event Fields**

- **eventid**
- **eventindex**
- **timestamp**
- **eventtype**
  - For example: “ANR”, “Error”

**Location Fields**

- **City**
- **Country**
- **Region**
  - For example: “California”, “Sichuan”
**Analytics Synthetic Sessions Data**

This page lists the field names available for Analytics Browser Synthetic Sessions data, along with the Events Service internal name. When accessing the data using ADQL queries, you need to use the Events Service internal name in the query. For details on the meaning of each metric, see Browser Synthetic Metrics.

**Event Type**: synth_session_records

**Key (event type identifier)**: Job Name (scheduleName)

<table>
<thead>
<tr>
<th>UI Field Name</th>
<th>Events Service Internal Name</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Session Fields</strong></td>
<td></td>
</tr>
<tr>
<td>Browser</td>
<td>browser</td>
</tr>
<tr>
<td></td>
<td>For example: “Internet Explorer”, “Chrome”</td>
</tr>
<tr>
<td>Browser Version</td>
<td>browserversion</td>
</tr>
<tr>
<td></td>
<td>For example: “52.0.2743.116”, “11.0.9600.18124”</td>
</tr>
<tr>
<td>City</td>
<td>geocity</td>
</tr>
<tr>
<td>Closed</td>
<td>closed</td>
</tr>
<tr>
<td>Country</td>
<td>geocountry</td>
</tr>
<tr>
<td>Devices</td>
<td>device</td>
</tr>
<tr>
<td></td>
<td>The computer form factor of the device. For example: “Computer” or “Mobile &amp; Tables”</td>
</tr>
<tr>
<td>Device Model</td>
<td>devicemodel</td>
</tr>
<tr>
<td></td>
<td>The model of the device. For example: “Galaxy S8” or “iPhone 7”</td>
</tr>
<tr>
<td>Device OS</td>
<td>deviceos</td>
</tr>
<tr>
<td></td>
<td>The operating system of the device. For example: “Windows” or “Android”</td>
</tr>
<tr>
<td>Duration (sec)</td>
<td>metrics.<code>Session Duration (ms)</code></td>
</tr>
<tr>
<td>End Time</td>
<td>endTimeMS</td>
</tr>
<tr>
<td>Error Type</td>
<td>failureType</td>
</tr>
<tr>
<td></td>
<td>For example: “Timed out”, “Test script crashed”</td>
</tr>
<tr>
<td>Job Name</td>
<td>scheduleName</td>
</tr>
<tr>
<td></td>
<td>The name of the synthetic job.</td>
</tr>
<tr>
<td>Region</td>
<td>georegion</td>
</tr>
<tr>
<td>Result</td>
<td>success</td>
</tr>
<tr>
<td>Schedule ID</td>
<td>measurementSpec.scheduleId</td>
</tr>
<tr>
<td>Session Type</td>
<td>sessionType</td>
</tr>
<tr>
<td></td>
<td>For example: “WEB_DRIVER”, “URL”</td>
</tr>
<tr>
<td>Session Visual Time (sec)</td>
<td>metrics.<code>Session Visual Time (ms)</code></td>
</tr>
<tr>
<td>Start Time</td>
<td>startTimeMS</td>
</tr>
<tr>
<td>Status</td>
<td>status</td>
</tr>
<tr>
<td>Timestamp</td>
<td>eventTimestamp</td>
</tr>
</tbody>
</table>
### Page View Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page Error Type</td>
<td>browserRecords.pageerror</td>
</tr>
<tr>
<td>RUM Page Name</td>
<td>browserRecords.pagename</td>
</tr>
<tr>
<td>URL</td>
<td>pageurl</td>
</tr>
</tbody>
</table>
## Analytics Connected Device Data

### ADQL Reference:

Connected Device Data provides details about each IoT connected device. The data collected includes information about devices, network requests and error events, and any custom properties and events that you instrument in your IoT application.

This page lists the required field names for Analytics Connected Device data, along with the Events Service internal name. When accessing the data using ADQL queries, you need to use the Events Service internal name in the query. Because IoT Monitoring largely depends on custom data, your connected device Analytics data will most likely include many more field names than given below.

Event Type: `iot_records`

Key (event type identifier): `appkey`

<table>
<thead>
<tr>
<th>UI Field Name</th>
<th>Events Service Internal Name</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Fields</strong></td>
<td></td>
</tr>
<tr>
<td>App Key</td>
<td>appkey</td>
</tr>
<tr>
<td>deviceguid</td>
<td>deviceguid</td>
</tr>
<tr>
<td>devicetype</td>
<td>devicetype</td>
</tr>
<tr>
<td>Timestamp</td>
<td>eventTimestamp</td>
</tr>
<tr>
<td>eventtype</td>
<td>eventtype</td>
</tr>
<tr>
<td><strong>customevent Fields</strong></td>
<td></td>
</tr>
<tr>
<td>eventsummary</td>
<td>customevent.eventsummary</td>
</tr>
<tr>
<td><strong>networkrequestevent Fields</strong></td>
<td></td>
</tr>
<tr>
<td>duration</td>
<td>networkrequestevent.duration</td>
</tr>
<tr>
<td>neterror</td>
<td>networkrequestevent.neterror</td>
</tr>
<tr>
<td>statuscode</td>
<td>networkrequestevent.statuscode</td>
</tr>
<tr>
<td>URL</td>
<td>networkrequestevent.url</td>
</tr>
<tr>
<td><strong>errorevent Field</strong></td>
<td></td>
</tr>
<tr>
<td>Resource Name</td>
<td>errorevent.name</td>
</tr>
</tbody>
</table>