Overview of Application Monitoring

After you learn the basics of AppDynamics, you're ready to learn how AppDynamics models application environments. The model serves as the framework around which AppDynamics organizes and presents performance information.

Application Model Overview

A typical application environment consists of the different components that interact in a variety of ways to fulfill requests from the application's users:

- Web applications served from an application server
- Databases or other data stores
- Remote services such as message queues and caches

AppDynamics app agents automatically discover the most common application frameworks and services. Using built-in application detection and configuration settings, agents collect application data and metrics to build flow maps.

A flow map visually represents the components of your application to help you understand how data flows among the application components. For example, the business transaction flow map for a simple e-commerce application below shows data flowing between web services, message queues, and databases:
Automatic detection lets you start exploring AppDynamics features quickly. As your understanding of AppDynamics matures and you identify areas unique to your environment, you can refine your application model.

### Business Transactions

In the AppDynamics model, a **business transaction** represents the data processing flow for a request, most often a user request. In real-world terms, many different components in your application may interact to provide services to fulfill the following types of requests:

- In an e-commerce application, a user logging in, searching for items or adding items to the cart
- In a content portal, a user requests content such as sports, business or entertainment news
- In a stock trading application, operations such as receiving a stock quote, buying or selling stocks

AppDynamics app agents discover requests to your application as entry points to a business transaction. Similar requests, such as user login, are treated as multiple instances of the same business transaction. The agents tag the request data and trace the request path as it passes from web servers to databases and other infrastructure components. AppDynamics collects performance metrics for each tier that processes the business transaction.

Because AppDynamics orients performance monitoring around business transactions, you can focus on the performance of your application components from the user perspective. You can quickly identify whether a component is readily available or if it is having performance issues. For instance, you can check whether users able to log in, check out or view their data. You can see response times for users, and the causes of problems when they occur.

### Business Applications

A **business application** is the top-level container in the AppDynamics model. A business application contains a set of related services and business transactions.

In a small AppDynamics deployment, only a single business application may be needed to model the environment. In larger deployments, you may choose to divide the model of the environment into several business applications.
The best way to organize business applications for you depends on your environment. A leading consideration for most cases, however, is to organize business applications in a way that reflects work teams in your organization, since role-based access controls in the Controller UI are oriented by business application.

Nodes

A node in the AppDynamics model corresponds to a monitored server or JVM in the application environment. A node is the smallest unit of the modeled environment. Depending on the agent type, a node may correspond to an individual application server, JVM, CLR, PHP application, Apache Web server.

Each node identifies itself in the AppDynamics model. When you configure the agent, you specify the name of the node, tier, and business application under which the agent reports data to the Controller.

Tiers

A tier is a unit in the AppDynamics model composed of a grouping of one or more nodes. How you organize tiers depends on the conceptual model of your environment.

Often a tier is used to a group of a set of identical, redundant servers. But that is not strictly required. You can group any set of nodes, identical or not, for which you want performance metrics to be treated as a unit into a single tier.

The single restriction is that all nodes in a single tier must be the same type. That is, a tier cannot have mixed types of agents, such as both .NET and Java nodes.

The traffic in a business application flows between tiers, as indicated by lines on the flow map, which are annotated with performance metrics.

In the AppDynamics model:

- There is no interaction among nodes within a single tier
- An application agent node cannot belong to more than one tier

Entities

An entity is any object that AppDynamics monitors, such as an application, tier, node, or even a business transaction. Entities typically have associated metrics, events, and a health status.

Historical and Live Entity Data

The Controller has an entity liveness module that tracks the "live" or "historical" status of the the four entity types application, tier, node, and business transaction for 365+ days.

- **Historical:** the oldest time (a year before the latest Controller restart) to the latest Controller restart time
- **Live:** the latest Controller restart time until the current time

Anchor Metrics for Entities

The entities have special metrics called anchor metrics that are used to determine the liveness of the entity. The table below lists the anchor metrics for each of the entities.

<table>
<thead>
<tr>
<th>Entity</th>
<th>Anchor Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Agent</td>
</tr>
<tr>
<td>Tier</td>
<td>Agent</td>
</tr>
<tr>
<td>Node</td>
<td>Agent</td>
</tr>
<tr>
<td>Business Transactions (BTs)</td>
<td>BTM</td>
</tr>
</tbody>
</table>
Liveness Status

The liveness of an entity affects the associated entities as the liveness is rolled up the hierarchy. If the entity type in the table below is live, you can determine the liveness of the associated entities in the right column.

<table>
<thead>
<tr>
<th>Live Entity</th>
<th>Liveness Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>An app is alive if any tiers in this app are alive</td>
</tr>
<tr>
<td>Tier</td>
<td>A tier is alive if any nodes in this tier are alive</td>
</tr>
<tr>
<td>Node</td>
<td>Any metrics from the particular node</td>
</tr>
<tr>
<td>Business Transactions (BTs)</td>
<td>BT metrics, Calls per Minute</td>
</tr>
</tbody>
</table>

How the Controller Displays Live Entities

Based on entity liveness status of the selected time range, the Controller will determine whether to count and display entities in the following places:

- flowmap
- Tier and Node list pages. This is also determined by whether the Performance Data checkboxes are selected. See Live Entity Data in Flowmaps.
- metric tree of the Metric Browser
- custom dashboards,
- AppDynamics REST APIs related to topology such as the Application Model API

Backends

A backend is a component that is not instrumented by an AppDynamics agent but that participates in the processing of a business transaction instance. A backend may be a web server, database, message queue, or another type of service.

The agent recognizes calls to these services from instrumented code (called exit calls). If the service is not instrumented and cannot continue the transaction context of the call, the agent determines that the service is a backend component. The agent picks up the transaction context at the response at the backend and continues to follow the context of the transaction from there.

Performance information is available for the backend call. For detailed transaction analysis in for the leg of a transaction processed by the backend, you need to instrument the database, web service or other application.

Integration with other AppDynamics Modules

This section describes how other AppDynamics APM Platform products work with Application Monitoring to provide complete, full visibility on application health and user experience.

Application Monitoring and Infrastructure Visibility

Infrastructure Visibility provides end-to-end visibility into the hardware and networks on which your applications run. You can use Infrastructure Visibility to identify and troubleshoot problems that affect application performance such as server failures, JVM crashes, and hardware resource utilization. There are three classes of Infrastructure Visibility functionality:

- You use the Machine Agent to collect basic hardware metrics. One Machine Agent license is included with each App Agent license that you purchase. You can deploy this Machine Agent only on the same machine where the App Agent is installed. The functionality provided by the Machine Agent includes:
  - Basic hardware metrics from the server OS. For example, %CPU and memory utilization, disk and network I/O
  - Custom metrics passed to the Controller by extensions
  - Run remediation scripts to automate your runbook procedures. You can optionally configure the remediation action to require human approval before the script is started.
  - Run JVM Crash Guard to monitor JVM crashes and optionally run remediation scripts
- If you have a Server Visibility license, the Standalone Machine Agent provides the following additional functionality:
• Extended hardware metrics such as machine availability, disk/CPU/virtual-memory utilization, and process page faults
• Monitor application nodes that run inside Docker containers and identify container issues that impact application performance
• The Tier Metric Correlator, which enables you to identify load and performance anomalies across all nodes in a tier
• Monitor internal or external HTTP and HTTPS services
• Group servers together so that health rules can be applied to specific server groups
• Define alerts that trigger when certain conditions are met or exceeded based on monitored server hardware metrics

• **Network Visibility** monitors traffic flows, network packets, TCP connections, and TCP ports. Network Agents leverage the APM intelligence of App Server Agents to identify the TCP connections used by each application. Network Visibility includes the following functionality:
  • Detailed metrics about dropped/retransmitted packets, TCP window sizes (Limited / Zero), connection setup/teardown issues, high round trip times, and other performance-impacting issues
  • Network Dashboard that highlights network KPIs (Key Performance Indicators) for tiers, nodes, and network links
  • Right-click dashboards for tiers, nodes, and network links that enable quick drill-downs from transaction outliers to network root causes
  • Automatic mapping of TCP connections with application flows
  • Automatic detection of intermediate load balancers that split TCP connections
  • Diagnostic mode for collecting advanced diagnostic information for individual connections

---

**Application Monitoring and Browser Real User Monitoring**

When you add End-User Monitoring to Application Performance Management, you can correlate business transaction performance to the user experience for those transactions. See [Correlate Business Transactions for Browser RUM](#).

If server app agents run on the applications that serve your browser applications, you can further configure the app server agents to inject JavaScript agent into the code that runs on the browser. Access the settings to configure injection in the Applications Configuration page. For more information, see [Automatic Injection of the JavaScript Agent](#) and [Assisted Injection](#).

**Application Monitoring and Database Visibility**

In Application Monitoring, a database called by an instrumented node is considered a remote service. You can get a significant amount of information on the interaction between the application node and database, but not from the database server perspective. When using Database Visibility with Application Monitoring, you can drill down to detailed database performance information directly from application flow maps. For more information see [Access Database Visibility from Application Monitoring Views](#).

**Application Monitoring and Analytics**

For those times when tracing application code does not provide enough clues to track down the cause of a problem, AppDynamics provides visibility into the transaction logs that can be correlated to specific business transaction requests. Log correlation visibility requires a license for both Transaction Analytics and Log Analytics. See [Business Transaction and Log Correlation](#).