PassQueen
a.k.a dumpsqueen.com

Cisco AppDynamics Associate Administrator (CAAA)

Cisco 500-425

Version Demo

Total Demo Questions: 8

Total Premium Questions: 81

Buy Premium PDF

https://passqueen.com

support@passqueen.com

Topic Break Down

Topic	No. of Questions
Topic 1, Cisco AppDynamics core functionality	46
Topic 2, Application performance monitoring	26
Topic 3, Database monitoring	9
Total	81

QUESTION NO: 1

Users are saying they are experiencing occasional slow response lime. You suspect a database issue and want to find snapshots showing slow database responsetimes Which two statements describe how to locate snapshots showing the slow database response? (Choose two.)

A. Go to the list of snapshots and apply a filter based on slow database response

B. Go to the dashboard for the database and select slowest database calls.

C. From the flow map, right-click the database icon and select view snapshots.

D. Go to the list of snapshots and select analyze for a group of snapshots.

ANSWER: A C

Explanation:

Go to the list of snapshots and apply a filter based on slow database response is correct because transaction snapshots contain detailed timing information for a business transaction, including time spent in exit calls such as database calls. The snapshot list can be filtered to isolate transactions whose database response time meets a specified threshold. This lets an administrator focus directly on transactions affected by slow database activity, while retaining the snapshot-level diagnostic details needed to investigate the call chain and associated execution context.

From the flow map, right-click the database icon and select view snapshots is also correct. A database shown on the flow map represents a backend dependency participating in transaction flow. Opening snapshots from that database context limits the investigation to transaction snapshots associated with calls to the selected database. The resulting snapshot data can then be reviewed or sorted using database-related timing information to identify the slow calls reported by users. Together, these approaches provide both a filter-driven workflow from the snapshot list and a dependency-driven workflow starting with the affected database on the flow map. Cisco AppDynamics documents transaction snapshots as diagnostic records for business transactions and describes using flow-map context to investigate application dependencies. See Transaction Snapshots and Application Flow Maps.

QUESTION NO: 2

What are three advantages of the custom dashboard feature? (Choose three.)

A. It makes drill down across tiers seamless

B. It turns data sharing on/off on the fly.

C. It monitors metrics of interest.

D. It finds the line of code having a performance issue

E. It schedules dashboard as a report

ANSWER: A C E

Explanation:

Custom dashboards provide a focused, shareable view of AppDynamics data by assembling selected metrics, health information, events, and other visual widgets onto one canvas. “It makes drill down across tiers seamless” is an advantage because a dashboard can present related application-flow and infrastructure information together, allowing an operator to move efficiently from an overall view to the relevant monitored entity and supporting detail. Dashboard layout and widget configuration make that high-level-to-detail workflow practical for operational teams.

“It monitors metrics of interest” is correct because dashboard widgets can be configured to show the specific application, business transaction, server, database, experience, or event data needed by a particular audience. This lets teams create concise views for monitoring service health, performance trends, and key operational indicators without requiring every user to navigate the full controller interface.

“It schedules dashboard as a report” is also an advantage. AppDynamics supports sharing dashboard results through reporting capabilities, including scheduled dashboard reports, so stakeholders can receive recurring visibility into the selected monitoring data. This is useful for regular operational reviews and communication with users who do not need continuous interactive access. See the AppDynamics documentation on Custom Dashboards and creating and managing custom dashboards and templates.

QUESTION NO: 3

What is used to capture application data in a single Business Transaction?

A. Data Collector

B. Windows Performance Counters

C. Information Point

D. MX Rules

ANSWER: A

Explanation:

Data Collector is correct because AppDynamics uses data collectors to extract and retain specific application data encountered during execution of an individual business transaction. A data collector is configured against a transaction context and identifies the source from which to obtain data, such as an HTTP request, method invocation, servlet session attribute, or other supported transaction property. It can then apply matching criteria and transformations before making the captured value available in the Controller. This lets an administrator add meaningful business context to transaction snapshots and diagnostics, such as an order identifier, customer segment, request parameter, or application-specific status value, without changing the core business-transaction naming and detection model. Captured values can support troubleshooting by associating relevant application data with the precise transaction execution in which it occurred, and can also be used in downstream AppDynamics capabilities where the collector type and agent configuration support that use. AppDynamics documentation describes business transactions as the central unit for monitoring application requests and explains how transaction-specific configuration provides visibility into their execution. See the AppDynamics Business Transactions documentation and the AppDynamics Data Collectors documentation.

QUESTION NO: 4

By default, which two Sensitive Data Filters substring does the Java Agent enable? (Choose two.)

A. substring "ssn"

B. substring "credit card"

C. substring "key"

D. substring "account"

E. substring "password"

ANSWER: C E

Explanation:

The Java Agent enables the default sensitive-data substring filters **substring "key"** and **substring "password"**. AppDynamics uses sensitive-data filtering to avoid collecting potentially confidential values into transaction snapshots and other Controller-visible diagnostic data. The default filters are evaluated against parameter or property names containing these terms, which helps protect common credential-related fields such as password parameters and key-based secrets. This behavior provides an initial privacy safeguard without requiring an administrator to define custom filters immediately after deploying an agent.

Administrators can review, add, modify, or remove sensitive-data filters as needed to align collection behavior with the organization's security and privacy requirements. Because applications may use organization-specific names for secrets or personal data, default filtering should be supplemented with carefully selected custom filters where appropriate. The configured filtering rules should also be validated in the context of the application's request data and diagnostic requirements. Cisco AppDynamics documents the Java Agent's sensitive-data filtering behavior and configuration under Filter Sensitive Data. For additional context on agent administration, see the Java Agent administration documentation.

QUESTION NO: 5

A customer wants a dashboard that will show them the number of current database connections on their application in a Timeseries graph, and compare it to past averages for the same time What option would solve this the fastest for the customer?

A. Create another dashboard and put them side by side.

B. Create a report that shows the historical data they are looking for

C. Include baseline data within widgets on the dashboard

D. Open up the application and change the time range to show the time they want

ANSWER: C

Explanation:

Include baseline data within widgets on the dashboard is the fastest solution because AppDynamics baselines are automatically calculated from the historical performance data collected for a metric. A baseline represents the expected normal range or average behavior for a comparable time period, allowing current measurements to be viewed in context rather than requiring the customer to manually locate and compare separate historical views. For a database-connections metric, a time-series widget can display the current connection count together with its baseline data. This gives the customer one graph that immediately shows whether the current number of connections follows the established pattern for that time or deviates from it. The approach directly meets both dashboard requirements: visualizing the current database-connection metric and comparing it with historical averages for the same period. It also uses AppDynamics' built-in Cognition Engine baseline capability, so it avoids creating additional dashboards, reports, or manually changing time ranges. Cisco AppDynamics documents baselines as the mechanism used to establish normal performance behavior and identify deviations over time. See the Baselines and Performance Trends documentation and the Custom Dashboards documentation.

QUESTION NO: 6

While troubleshooting a performance issue on a Java application the engineer determines there is a possible memory leak in the JVM Using AppDynamics, how would the engineer determine rf there is a memory leak?

A. Examine the values on the Server tab on one of the affected Nodes.

B. Configure Object Instance Tracking on the Tier in question.

C. Verify and adjust the Memory Monitoring configuration for the Tier in question

D. Analyze the information on the Memory tab on one of the affected Nodes

ANSWER: D

Explanation:

Analyze the information on the Memory tab on one of the affected Nodes. In AppDynamics, the Java Agent exposes JVM memory-health data at the node level, and the Memory tab is specifically designed to investigate heap behavior and potential memory leaks. An engineer can review memory-pool utilization, garbage-collection activity, and the trend of retained objects or collections over time. A sustained increase in memory consumption after garbage collection is an important signal that objects may be unintentionally retained.

For supported Java Agent configurations, AppDynamics Automatic Leak Detection identifies collections that are both actively used and continuously growing. The engineer can drill into a suspected collection to view its contents and access traces. These traces help connect retained objects to the relevant application code path and business transactions, providing evidence needed to validate the suspected leak and begin remediation. This workflow is centered on the affected node's Memory tab because it combines JVM memory trends with leak-detection analysis in the context of the instrumented application. See the AppDynamics documentation on Java Memory Leaks and monitoring the JVM.

QUESTION NO: 7

The applications and servers are running normally What are two important reasons for the engineer to view and analyze server performancedata at this time? (Choose two.)

A. Visualizing application performance during normal operations helps identify when the application performance is abnormal.

B. Graphs and charts collected at this time are useful for showing application owners the stability of their application

C. Collecting Server metric performance during normal operations helps determine code problems

D. Collecting the errors during this period will identify memory problems in the application.

ANSWER: A B

Explanation:

Visualizing application performance during normal operations helps identify when the application performance is abnormal because normal-operation measurements establish a practical baseline for server health. AppDynamics Server Monitoring collects infrastructure metrics such as CPU utilization, memory use, disk activity, network activity, and process-level resource consumption. Reviewing these metrics while systems are healthy lets an engineer understand expected patterns, including normal peaks and recurring workload cycles. That context makes meaningful deviations easier to recognize later, supporting faster identification of abnormal resource behavior that may affect an application.

Graphs and charts collected at this time are useful for showing application owners the stability of their application because AppDynamics dashboards and metric visualizations provide an understandable historical view of server behavior and availability. A stable trend across an appropriate time range can be shared with stakeholders to demonstrate that the infrastructure supporting the application has operated consistently under normal demand. These visualizations also give application owners a common baseline for future performance discussions and capacity planning. Cisco AppDynamics describes Server Monitoring and its server-health metrics in its Server Monitoring documentation; its Server Metrics documentation details the monitored resource data used for this analysis.

QUESTION NO: 8

A Java application returns an order category from a method, and the administrator needs that value displayed in transaction snapshots for a selected Business Transaction. Which configuration is most appropriate?

A. Create a Method Invocation Data Collector that captures the method return value

B. Create an HTTP Data Collector that captures a response header

C. Create a health rule based on Business Transaction response time

D. Create a custom dashboard widget for the Business Transaction

ANSWER: A

Explanation:

A Method Invocation Data Collector can capture application data from a selected Java class and method, including a method return value. The administrator can scope the collector to the relevant Business Transaction and select the returned order-category value for collection.

An HTTP Data Collector is appropriate for HTTP parameters, headers, or cookies rather than a Java method return value. A health rule evaluates metric conditions, while a custom dashboard visualizes existing monitoring data; neither extracts the method value from the application execution.