PassQueen
a.k.a dumpsqueen.com

Certified Implementation Specialist - Event Management Exam

ServiceNow CIS-EM

Version Demo

Total Demo Questions: 15

Total Premium Questions: 154

Buy Premium PDF

https://passqueen.com

support@passqueen.com

Topic Break Down

Topic	No. of Questions
Topic 1, Event Management Overview	48
Topic 2, Event Management Setup	17
Topic 3, Event Management Administration	60
Topic 4, Event Management Integrations	27
Topic 5, Mix Questions	2
Total	154

QUESTION NO: 1 - (HOTSPOT)

HOTSPOT

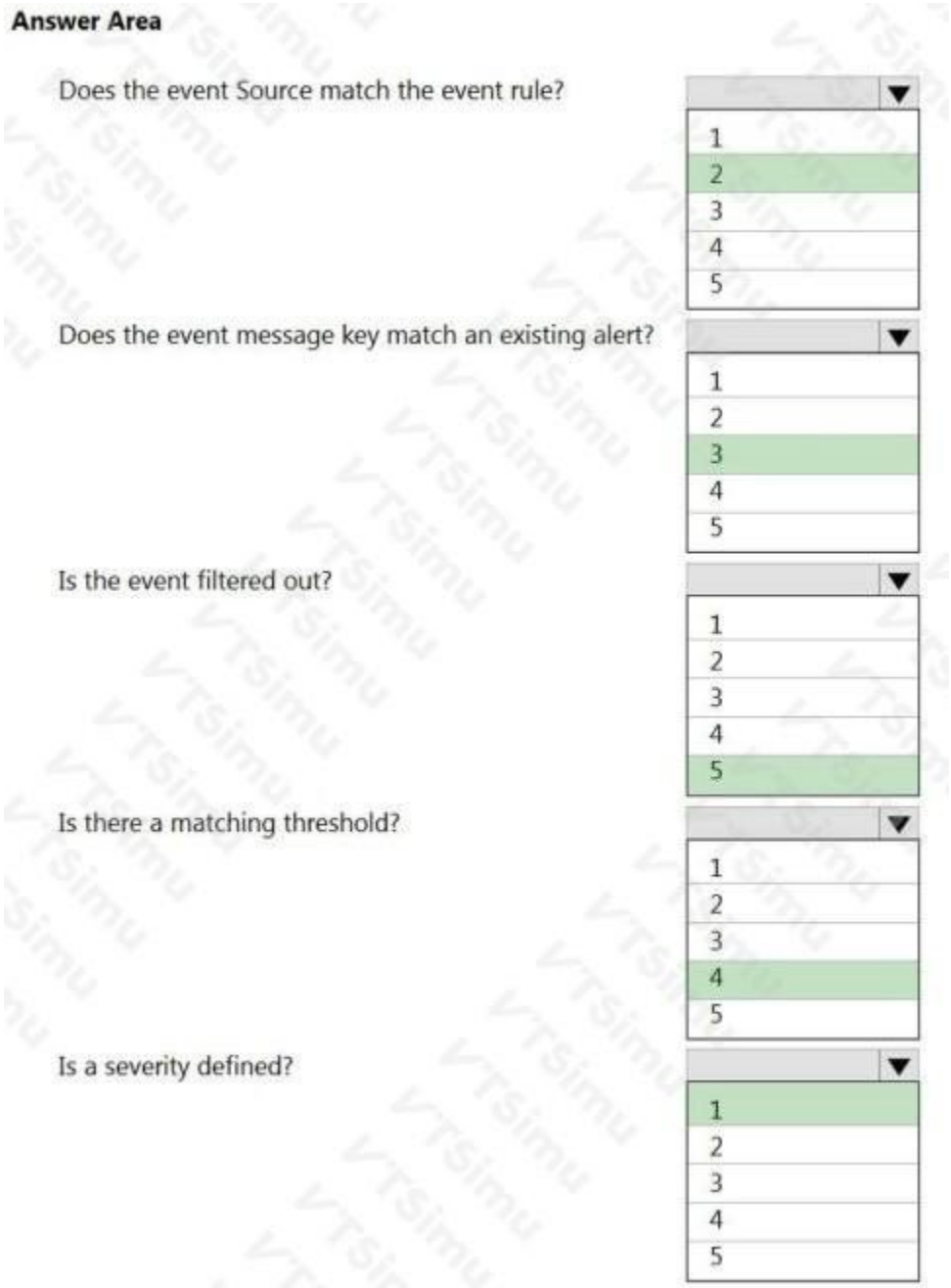
In what sequence are events processed?

Hot Area:

Answer Area

ANSWER:

Answer Area



Explanation:

ServiceNow Event Management processes the event by first establishing that the event has a severity. Severity is a foundational event attribute because it provides the operational significance used as the event moves into alert handling. The processor then evaluates whether the event source matches an event rule. A matching rule provides the applicable processing context for the incoming event and enables the platform to apply the intended event-management behavior.

Next, the platform uses the event message key to determine whether the incoming event belongs to an alert that already exists. This supports alert updating and deduplication, preventing repeated events representing the same condition from unnecessarily producing separate alerts. After that correlation step, Event Management evaluates whether a matching threshold applies. Threshold logic supports event aggregation and determines whether the defined volume or rate condition has been met for alert processing.

The filtering decision occurs last in this sequence. Accordingly, the correct order is: a severity is defined first, the event source matches an event rule second, the event message key matches an existing alert third, a matching threshold is identified fourth, and the event is filtered out fifth. The highlighted values in the image express exactly that order. ServiceNow's official documentation hub for Event Management and IT Operations Management is available at ServiceNow Product Documentation.

QUESTION NO: 2

Which two statements describe why a stable Message Key is important for an Event Management integration? (Choose two.)

A. It supports event de-duplication

B. It correlates subsequent updates with an existing alert

C. It identifies the assignment group for an alert

D. It determines the alert severity

E. It replaces the need for a Source value

ANSWER: A B

Explanation:

A stable **Message Key** allows Event Management to identify repeated notifications for the same underlying condition and de-duplicate the event records. It also allows subsequent events, including clear events, to be correlated with the appropriate existing alert rather than creating a new alert for each update.

The Message Key should remain consistent while the monitored condition remains the same. A value that changes with every notification, such as a timestamp-based value, prevents dependable event and alert correlation. See the ServiceNow Event fields reference.

QUESTION NO: 3

You have a very large networking environment and have noticed that your event notifications are either not being triggered or are delayed.

What are best options to try to resolve this issue? (Choose two.)

A. Ensure all Event Management – process events jobs are set to a Ready state

B. Verify that the Bucket field in the event table is set to zero (0)

C. Add additional event processor jobs

D. Ensure multi-node event processing is disabled

ANSWER: A C

Explanation:

Ensure all Event Management – process events jobs are set to a Ready state is correct because these scheduled worker jobs are responsible for obtaining and processing incoming event records. A worker that is not ready to run cannot contribute to the event-processing workload, which can cause events to remain unprocessed and delay the downstream alerting and notification flow. Confirming that the event-processing jobs are available and ready is therefore an essential operational check when event volume is high.

Add additional event processor jobs is also correct. Event Management processes events asynchronously, and a very large network can generate a sustained event rate that exceeds the capacity of the currently configured workers. Increasing the number of processor jobs provides additional parallel processing capacity, helping reduce the queue backlog and allowing qualifying events to be evaluated and acted on promptly. This scaling action should be accompanied by monitoring of platform and node capacity so that the additional workers can operate effectively. ServiceNow's Event Management documentation describes event processing and the operational configuration of Event Management, while scheduled-job documentation explains the role and execution state of scheduled jobs: ServiceNow Event Management documentation and ServiceNow scheduled jobs documentation.

QUESTION NO: 4

What actions can be configured in an Event Management event rule? (Choose three.)

A. Set event field values

B. Run a script

C. Drop an event

D. Update a change request state

E. Approve an incident

ANSWER: A B C

Explanation:

Event rules process incoming events and can transform event data before alert processing. An event rule can set or modify event field values, execute JavaScript for more complex transformation logic, and drop events that should not proceed through Event Management processing.

These actions allow an implementation to normalize vendor-specific data, enrich incoming events, and reduce monitoring noise before alerts are created. Alert-level actions, such as assignment and incident creation, are handled after alert creation by alert management rules or other alert-management processes. See the ServiceNow Event Rules documentation.

QUESTION NO: 5

Which are recommended best practices for Event Management? (Choose three.)

A. Filter out events on ServiceNow Instance for easier consolidation and aggregation.

B. Promote all events to alerts during initial implementation until you fully understand which should be ignored.

C. Filter out events at source rather than in the ServiceNow instance.

D. Base-line “normal-state” events to filter out background noise.

E. Ignore all non-critical events during initial implementation to streamline processing; add alerts over time as time and resources allow.

ANSWER: C D E

Explanation:

Event Management implementations are most effective when they reduce noise before it reaches the platform and introduce monitoring coverage in a controlled, iterative way. “Filter out events at source rather than in the ServiceNow instance.” is recommended because source-side filtering prevents unnecessary event traffic and processing while retaining only signals that have operational value. “Base-line “normal-state” events to filter out background noise.” is also a sound practice: understanding the normal operating pattern of a service or infrastructure component enables teams to distinguish meaningful deviations from expected, repetitive events.

“Ignore all non-critical events during initial implementation to streamline processing; add alerts over time as time and resources allow.” supports a phased adoption model. Initial Event Management deployments should focus on high-value, actionable signals and critical service-impacting conditions. Teams can then refine rules, thresholds, event filters, alert correlation, and monitoring scope as they gain confidence in the quality and usefulness of the data. This approach helps avoid overwhelming operators with low-priority signals and allows the implementation to mature based on operational feedback. ServiceNow describes Event Management as collecting events from monitoring tools, processing them, and identifying actionable alerts; effective event filtering and noise reduction are central to that outcome. See ServiceNow Event Management overview and ServiceNow IT Operations Management.

QUESTION NO: 6

During processing of the event and if the event Severity is blank, the state of the event is set to:

A. Ready

B. Ignored

C. Error

D. Processing

ANSWER: C

Explanation:

Error is correct because severity is a required event attribute for Event Management processing. During event normalization and processing, ServiceNow uses severity to classify the event's importance and to support downstream event-handling logic, including alert creation, aggregation, and operational prioritization. If an incoming event does not contain a severity value, Event Management cannot complete the required validation successfully. The platform therefore sets the event record's state to **Error**, signaling that the event has failed processing and requires correction at the source or in the inbound integration's transformation or mapping logic.

This behavior helps protect the quality of event data entering the instance. Integration owners should ensure that their event source supplies a supported severity value, or configure an appropriate mapping that converts source-specific priority values into ServiceNow Event Management severity values before the event is processed. Correcting the missing field and resending a valid event enables normal processing. ServiceNow's event integration requirements document identifies the required event fields and the processing behavior for invalid or incomplete events: Event integration requirements. Additional Event Management documentation is available at ServiceNow Event Management documentation.

QUESTION NO: 7

What two key steps must be performed after creating a new connector instance? (Choose two.)

A. Assign a MID Server to the connector

B. Enter credentials for the connector

C. Debug the connector

D. Test the connector

E. Activate the connector

ANSWER: D E

Explanation:

After a connector instance is created and its required connection details have been configured, it should be tested and then activated. **Test the connector** verifies that the Event Management connector can communicate with the external event source using the instance's configuration. This validation is important because it confirms connectivity and helps ensure the connector can successfully retrieve or receive event data before it is put into operational use. Testing provides an administrator with an opportunity to identify and resolve configuration or communication issues while the connector is not yet enabled for normal event processing.

Activate the connector enables the configured connector instance so that it can begin its intended operational function in Event Management. Once active, the connector can process events from its associated external source and make those events available to Event Management for normalization, alert creation, and subsequent event handling workflows. Performing the test before activation supports a controlled implementation process: validate the integration first, then enable it for production event flow. ServiceNow's connector-instance configuration guidance identifies testing and activation as the post-creation actions for a connector instance. See the ServiceNow connector instance configuration documentation and the ServiceNow Event Management documentation.

QUESTION NO: 8

Which three event attributes should be consistently populated to support dependable event processing and service-aware alert handling? (Choose three.)

A. Source

B. Message Key

C. Node

D. Connector polling interval

E. Incident number

ANSWER: A B C

Explanation:

A valid **Source** identifies the monitoring system or integration that submitted the event and is required for event processing. A stable **Message Key** supports de-duplication and allows later events to update the same operational condition. A normalized **Node** supports CI binding by providing an identifier that Event Management can match to a configuration item.

Using consistent values for these attributes improves event quality, alert correlation, and CMDB context. Polling frequency can affect collection latency, but it does not provide event identity or CI association. See the ServiceNow Event fields reference.

QUESTION NO: 9

Within an event rule, how would you parse a nodename out of your raw event data?

A. JavaScript

B. Groovy script

C. PowerShell script

D. Regex statement

ANSWER: A

Explanation:

JavaScript is the appropriate mechanism for parsing a nodename from raw event data within an Event Management event rule. Event rules can use script-based transformations to inspect an incoming event's raw payload or source fields, extract the needed value, and assign that value to the event's `node` field. This is useful when the nodename is embedded in unstructured text, a JSON-like payload, or a vendor-specific field rather than arriving directly in the normalized node field. A JavaScript expression can apply string methods, pattern matching, conditional logic, and value cleanup before setting the normalized event attribute. Normalizing the node is important because Event Management uses event field values for processing, correlation, alert creation, and CMDB-related context. The script should reliably return only the intended nodename and account for expected payload variations so that events from the same managed entity are identified consistently. ServiceNow documents event rules as the mechanism for transforming and processing incoming events before alert handling, and its server-side scripting documentation describes the JavaScript environment used for platform logic. See ServiceNow Event Rules documentation and ServiceNow server-side scripting documentation.

QUESTION NO: 10

What ServiceNow feature would you configure to process incoming email to create events?

A. Event field mapping

B. Event processing jobs

C. Transforms

D. Event Filler

E. Inbound actions

ANSWER: E

Explanation:

Inbound actions are the ServiceNow feature used to inspect and process received email messages. An inbound email action runs when an email reaches the instance and meets its configured conditions, such as a particular recipient, sender, subject pattern, or message content. Its action script can extract the relevant alert details from the email body and headers, then create and populate an Event Management event record in the `em_event` table. This makes inbound actions appropriate when an external monitoring source delivers alerts by email rather than through a direct Event Management connector, API, or MID Server integration.

The inbound-email framework supplies useful objects and properties to the action, including the incoming email content and recipient information, so the implementation can normalize fields such as source, node, metric name, severity, message key, description, and additional information before inserting the event. After the event is created, Event Management applies its normal event processing behavior, including event rules, alert creation, correlation, and remediation workflows where configured. Configure the inbound action with clear conditions and a script appropriate to the monitoring email format. See the ServiceNow guidance on creating inbound email actions and the Event Management documentation.

QUESTION NO: 11

The ServiceNow standard and shared set of service-related definitions that enable and support true service level reporting is known as what?

A. Service level data model

B. Business service data model

C. Application service data model

D. Common service data model

ANSWER: D

Explanation:

Common service data model is correct because the Common Service Data Model (CSDM) is ServiceNow's standardized framework and shared vocabulary for defining services and the relationships among business applications, application services, technology services, service offerings, and related configuration items. By establishing consistent service-related definitions and relationships in the CMDB, CSDM makes it possible to associate operational events, incidents, changes, and performance data with the services actually consumed by users and customers.

This common model is essential for meaningful, service-aware reporting. It provides the structure needed to measure service health and availability at the appropriate service and service-offering level, rather than reporting only on isolated infrastructure components. It also supports consistent ownership, impact analysis, and service-level reporting across teams, which is why ServiceNow describes CSDM as the common framework for service-related data. The official CSDM documentation explains its domains, model relationships, and adoption guidance: ServiceNow Common Service Data Model documentation. ServiceNow's CMDB overview also describes how accurate configuration and relationship data supports service visibility and operational decisions: What is a CMDB?

QUESTION NO: 12

When are anomaly alerts generated by Operational Intelligence displayed in alert intelligence?

A. When the statistical model threshold is breached

B. When they are promoted to IT alerts

C. When it is manually promoted in insights explorer

D. When the anomaly score is greater than 100

ANSWER: B

Explanation:

Operational Intelligence continuously analyzes time-series metric data and can create anomaly alerts when observed behavior deviates significantly from its learned baseline. These anomaly alerts initially represent detected metric anomalies and are available for investigation in the Operational Intelligence experience, including Insights Explorer. They are not automatically part of the Event Management alert stream merely because an anomaly was detected.

An anomaly alert is displayed in Alert Intelligence after it is promoted to an IT alert. Promotion creates the Event Management alert record and makes the anomaly available alongside other operational alerts for triage, correlation, and remediation workflows. This separation allows teams to review anomaly findings and decide which ones warrant operational action, reducing unnecessary noise in Alert Intelligence while preserving the underlying anomaly analysis.

ServiceNow documents this workflow in its Operational Intelligence guidance for viewing metric anomaly alerts: View metric anomaly alerts. Additional product information is available in the ServiceNow Operational Intelligence overview.

QUESTION NO: 13

Which is a correct regex expression to capture only the server's hostname (webserver3) in "The server webserver3.domain.com is down."?

A. The server (.*)\..*

B. .*(\w+\Aw+VAw+).*

C. The server (.*)\s.*

D. The server (.*?)\„*

E. The server ([^\.]+)\..*

ANSWER: E

Explanation:

`The server ([^\.]+)\..*` is correct because it first matches the fixed text `The server` , then captures the hostname in a parenthesized capture group. The character class `[^\.]` means “any character except a period,” and the `+` quantifier consumes one or more of those characters. Consequently, the capture group stops at the first period following `webserver3`, so the captured value is exactly `webserver3`. The escaped period `\.` then matches the literal dot separating the hostname from the DNS domain, while `.*` consumes the remaining domain and sentence text. This approach is appropriate when the desired value is the first DNS label rather than the entire fully qualified domain name. Capturing groups are the standard regex mechanism for retaining a selected portion of an otherwise larger match; see the MDN documentation on capturing groups. The use of a negated character class to stop at a delimiter is also consistent with standard regular-expression syntax described in the Java Pattern API documentation.

QUESTION NO: 14 - (SIMULATION)

In what sequence are events processed?

ANSWER: See the explanation for the answer

Explanation:

Event processing sequence:

1. **Event filter rules** — discard events that should not be processed.
2. **Event field-mapping rules** — map incoming/source fields into ServiceNow event fields.
3. **Event transformation rules** — modify or normalize the mapped event values.
4. **Event alert rules** — use the processed event to create or update an alert (including event/alert de-duplication behavior).

After an alert exists, downstream alert processes, such as alert correlation and alert-management rules, can operate on that alert. They are not earlier event-processing stages.

QUESTION NO: 15

What is the preferred method of parsing in the Transform/Compose step of an event rule?

A. Python

B. Regex

C. sed/awk

D. JavaScript

ANSWER: B

Explanation:

Regex is the preferred parsing method in the Transform/Compose step because Event Management event rules are designed to use regular-expression matching to identify and extract values from incoming event fields. A regular expression can match structured or semi-structured text and capture the relevant portions of an event payload, such as a host name, metric, severity, interface, or application identifier. The captured values can then be mapped into event fields or used while composing output values, allowing one rule to normalize similarly formatted events consistently.

Using regex directly in the event-rule transform workflow also provides a declarative, maintainable approach that aligns with the platform's event-processing design. The expression must be constructed to match the event data as required by the rule; when the expected pattern is not matched, the parsing operation cannot extract the intended values. This makes careful pattern testing important when configuring rules for events received from monitoring tools. ServiceNow's Event Management documentation describes composing event-rule output and the use of regular expressions for parsing and transforming event content. See ServiceNow: Compose event rule output and ServiceNow Event Management documentation.