DUMPSARENA

Splunk Certified Developer Exam

Splunk SPLK-2001

Version Demo

Total Demo Questions: 10

Total Premium Questions: 102

Buy Premium PDF

https://dumpsarena.co

sales@dumpsarena.co

sales@dumpsarena.co
dumpsarena.co

Topic Break Down

Topic	No. of Questions
Topic 1, Splunk Development	33
Topic 2, Splunk App Development	39
Topic 3, Custom Search Commands	1
Topic 4, Custom Visualizations	4
Topic 5, Custom REST Endpoints	24
Topic 6, Mix Questions	1
Total	102

QUESTION NO: 1

When updating a knowledge object via REST, which of the following are valid values for the sharing Access Control List property?

A. App

B. User

C. Global

D. Nobody

ANSWER: A

Explanation:

App is a valid value for the `sharing` property when a knowledge object's access-control list is updated through Splunk's REST API. Setting sharing to `app` makes the knowledge object available within its current app context, subject to the permissions assigned through the ACL. This is useful for objects such as saved searches, event types, field extractions, and macros that should be reusable by users of one app without being exposed across every app in the Splunk deployment.

In REST requests, ACL settings are managed using the object's `/acl` endpoint. The `sharing` field determines the scope in which Splunk exposes the object, while other ACL fields, including owner and role-based read/write permissions, determine who can access or modify it. App-level sharing is therefore a standard way to package and maintain knowledge objects that support an application's searches, dashboards, and workflows. Splunk's REST API documentation describes access-control endpoints and the ACL properties used to update knowledge objects. See the Splunk REST API User Manual and the knowledge object permissions documentation.

QUESTION NO: 2

Which files within an app contain permissions information? (Select all that apply.)

A. local/metadata.conf

B. metadata/local.meta

C. default/metadata.conf

D. metadata/default.meta

ANSWER: B D

Explanation:

Within a Splunk app, permissions and sharing settings are stored in metadata files named `default.meta` and `local.meta`, located in the app's `metadata` directory. Therefore, `metadata/default.meta` and `metadata/local.meta` contain permissions information. These files define access-control metadata for app knowledge objects, including settings such as object ownership, sharing scope, and role-based read or write access. The `default.meta` file supplies the app's packaged default permissions, while `local.meta` holds local permission customizations that can override the defaults without modifying the app's distributed configuration. This separation follows Splunk's standard configuration-layering model and helps preserve local administrative changes during app upgrades. Splunk documents `default.meta` as the configuration file for default metadata and ACL definitions; its counterpart, `local.meta`, is the local-layer metadata file used for site-specific settings. See Splunk's default.meta configuration reference and its guidance for managing access to custom search commands.

QUESTION NO: 3

Which of the following ensures that quotation marks surround the value referenced by the token?

A. $token_name|s$

B. “$token_name$”

C. ($token_name$)

D. \“$token_name$\”

ANSWER: A

Explanation:

`$token_name|s$` is the correct token syntax because the `|s` token filter applies Splunk search-string escaping to the token value. This filter places double quotation marks around the resolved value and escapes any double quotation marks that occur within that value. As a result, the expanded token is handled as one quoted search-string value when it is inserted into a search, which is especially important for values containing spaces or characters with significance in SPL.

For example, if the token resolves to `error message`, the filtered expansion is inserted as `"error message"`, preserving the complete value as a single string in the resulting search. Token filters are applied by appending a pipe and filter name inside the token delimiters. Splunk documents the search-string filter as the appropriate filter when a token value must be safely represented as a quoted search literal. This behavior supports reliable dynamic dashboard searches while ensuring that the token’s value is interpreted in the intended quoted context. See Splunk’s Token usage and token filters documentation and the dashboard token reference.

QUESTION NO: 4

Which of the following are reserved field names in a KV Store? (Select all that apply.)

A. _key

B. _time

C. _user

D. _source

ANSWER: A B C

Explanation:

`_key`, `_time`, and `_user` are reserved field names for Splunk KV Store collections. The `_key` field is the unique identifier for each KV Store record. It is fundamental to record addressing and is used when retrieving, updating, or deleting a specific record through KV Store interfaces. Splunk assigns a key automatically when one is not supplied, so application developers should treat this field as system-managed identity data.

The `_user` field is reserved for ownership-related behavior. It is particularly relevant to user-scoped KV Store collections, where Splunk uses the field to associate records with the user context. The `_time` field is also reserved by the KV Store implementation and must not be repurposed as an application-defined field. When designing a collection schema, use distinct names such as `event_time`, `owner`, or an application-specific identifier for custom data instead. This avoids collisions with Splunk-managed metadata and preserves expected KV Store behavior. Splunk documents these reserved fields in its KV Store collections developer documentation and collection configuration details in the collections.conf reference.

QUESTION NO: 5

Which of the following statements are true of search macros? (Select all that apply.)

A. Macro definitions are stored in macros.conf.

B. Macros can accept arguments.

C. Macros expand at search time.

D. Macros are intended to store encrypted credentials.

ANSWER: A B C

Explanation:

Search macro definitions are stored in `macros.conf`. A macro can contain reusable SPL that is expanded when the macro is invoked in a search. This enables developers to standardize commonly used search expressions and update the shared definition without editing every search that uses it.

Macros can also be defined with arguments, allowing callers to pass values that are substituted into the macro definition. Macros expand at search time and should not be used as a mechanism for storing sensitive credentials, because macro definitions can be visible to users who have access to the associated knowledge object. See the search macros documentation and the macros.conf specification.

QUESTION NO: 6

Which of the following is an example of a valid syntax for specifying an absolute time range modifier in a search?

A. earliest=01/01/2019:00:00:00

B. earliest=01/01/2019T00:00:00

C. earliest=2019-01-01 00:00:00

D. earliest=2019-01-01T00:00:00

ANSWER: A

Explanation:

`earliest=01/01/2019:00:00:00` is valid syntax for an absolute time-range modifier in Splunk SPL. The `earliest` modifier limits the search to events whose timestamps are at or after the supplied time. For an absolute time, Splunk accepts the date-and-time form `MM/DD/YYYY:HH:MM:SS`; the colon between the date and time is an essential part of this documented format. In this example, the search begins at midnight on January 1, 2019.

Time modifiers such as `earliest` and `latest` can be included with the base search to constrain the time interval before search commands process matching events. This is generally preferable to filtering on `_time` later in the pipeline because the time range can be applied by Splunk during search initialization. If no time zone is explicitly supplied, interpretation of an absolute timestamp follows the applicable Splunk/user time-zone context. Splunk's documentation distinguishes these fixed absolute timestamps from relative expressions such as `-24h` or `@d`, which are calculated relative to a reference time. See Splunk's Search time modifiers reference and its guide to specifying time modifiers in searches.

QUESTION NO: 7

Which of the following is a way to monitor app performance? (Select all that apply.)

A. Using Splunk logs.

B. Using the search job inspector.

C. Using the Monitoring Console.

D. Using the storage/collections/config REST endpoint.

ANSWER: A B C

Explanation:

Using Splunk logs, using the search job inspector, and using the Monitoring Console are all valid ways to monitor the performance of a Splunk app and the Splunk services on which it depends. Splunk's internal logs, including data in the `_internal` index, record operational messages, warnings, errors, search activity, and component-level behavior. Developers can search these logs to identify symptoms such as failed scripted inputs, lookup issues, dashboard errors, or slow-running searches.

The search job inspector is particularly useful when an app's searches or dashboards are slow. It exposes detailed timing and execution metrics for an individual search job, helping a developer understand where time was spent during parsing, index scanning, command execution, result generation, and related search phases. Splunk documents these inspection details at Inspect search jobs.

The Monitoring Console provides deployment-level health and performance views. Its dashboards and metrics help administrators and developers observe search performance, indexing throughput, resource utilization, and distributed-search health, all of which can affect an app's user experience. See Splunk's Monitoring Console documentation for its monitoring capabilities.

QUESTION NO: 8

What application security best practices should be adhered to while developing an app for Splunk? (Select all that apply.)

A. Review the OWASP Top Ten List.

B. Store passwords in clear text in .conf files.

C. Review the OWASP Secure Coding Practices Quick Reference Guide.

D. Ensure that third-party libraries that the app depends on have no outstanding CVE vulnerabilities.

ANSWER: A C D

Explanation:

Reviewing the OWASP Top Ten List is an important practice because it identifies widely recognized categories of web-application risk, such as injection, broken access control, and security misconfiguration. Splunk app developers should use these risks as a practical baseline when designing, coding, testing, and reviewing an app. The OWASP Top 10 provides the industry context needed to identify common weaknesses before an app is deployed.

Reviewing the OWASP Secure Coding Practices Quick Reference Guide complements that high-level risk awareness with actionable secure-development guidance. It covers practices for areas such as input validation, authentication, session management, error handling, and data protection, all of which are relevant when an app processes user-controlled input or interacts with Splunk services.

Ensuring that third-party libraries that the app depends on have no outstanding CVE vulnerabilities is also a core dependency-security practice. App code can be secure while a bundled or required dependency introduces a known exploitable flaw. Developers should inventory dependencies, monitor vulnerability disclosures, update to remediated versions, and remove or mitigate vulnerable components before release. Splunk's app-security guidance recommends applying established secure-coding practices, while OWASP provides the associated risk and coding references. See Splunk app security best practices and the OWASP Top 10.

QUESTION NO: 9

Which of these URLs could be used to construct a REST request to search the employee KV store collection to find records with a rating greater than or equal to 2 and less than 5?

A.
'http://localhost:8089/servicesNS/nobody/search/storage/collections/data/employees?query={$and:[{rating:{$gte:2}},{rating:{$lt:5}}]} &output_mode-json'

B.
'http://localhost:8089/servicesNS/nobody/search/storage/collections/data/employees?query={$and:[{rating:$gte:2}},{rating:{$lt:5}}]} &output_mode=json'

C.
'http://localhost:8089/servicesNS/nobody/search/storage/collections/data/employees?query={%22rating%22:{%22$gte%22:2}},{%22$and%22},{% 22rating%22:{%22$lt%22:5}}}&output_mode=json'

D.
'http://localhost:8089/servicesNS/nobody/search/storage/collections/data/employees?query={%22$and%22:[{%22rating%22:{%22$gte%22:2}},{%
22rating%22:{%22$lt%22:5}}]}&output_mode=json'

ANSWER: D

Explanation:

The URL `'http://localhost:8089/servicesNS/nobody/search/storage/collections/data/employees?query={%22$and%22:[{%22rating%22:{%22$gte%22:2}},{%22rating%22:{%22$lt%22:5}}]}&output_mode=json'` correctly targets the KV store collection-data REST endpoint for the `employees` collection in the `search` app's global (`nobody`) namespace. The collection-data endpoint accepts a `query` parameter containing a MongoDB-style query expression. Here, `$and` combines two conditions applied to the same `rating` field: `$gte: 2` includes ratings of 2 or above, and `$lt: 5` limits the result to ratings strictly below 5. Together, these conditions return records whose rating is in the interval from 2 (inclusive) through 5 (exclusive).

The double quotation marks required by the JSON query are URL encoded as `%22`, allowing the JSON expression to be transmitted safely as a query-string parameter. Finally, `output_mode=json` requests a JSON response, which is appropriate for programmatic REST clients. Splunk documents the KV store collection-data endpoint and its query capabilities in the KV store REST API reference; see also the Splunk REST API documentation for REST request conventions.

QUESTION NO: 10

Which of the following options would be the best way to identify processor bottlenecks of a search?

A. Using the REST API.

B. Using the search job inspector.

C. Using the Splunk Monitoring Console.

D. Searching the Splunk logs using index=â€œinternalâ€□.

ANSWER: B

Explanation:

Using the search job inspector is the best way to identify processor bottlenecks in an individual search because it exposes detailed, job-specific execution metrics. In Splunk Web, the Job Inspector breaks the search into its execution phases and reports timing, resource usage, and diagnostic information for the completed search job. Its performance information helps a developer determine where the search spent time, including time associated with search processing and command execution. This makes it practical to isolate expensive portions of SPL, such as commands or processing stages that consume disproportionate CPU time or otherwise delay completion.

The Job Inspector also provides counters and properties that add context to performance analysis, including event and result counts, scan-related metrics, search duration, and search log details. Reviewing these data points together enables a developer to identify the portion of a search that should be optimized, then validate whether changes to filtering, command order, field extraction, data-model use, or aggregation improve execution. Splunk documents the Job Inspector as the interface for viewing information about a specific search job and diagnosing search performance. See Splunk Job Inspector documentation and Splunk guidance for optimizing searches.