DUMPSARENA

Alfresco Process Services Certified Engineer (APSCE)

Alfresco APSCE

Version Demo

Total Demo Questions: 8

Total Premium Questions: 83

Buy Premium PDF

https://dumpsarena.co

sales@dumpsarena.co

sales@dumpsarena.co
dumpsarena.co

QUESTION NO: 1

Why would a developer create a custom BPMN editor stencil? Choose 2 answers

A. To make custom form controls available to a Process Designer

B. To make custom process elements available to a Process Designer

C. To make predefined forms available to a Process Designer

D. To limit the process elements that are available to a Process Designer

E. To limit the form controls that are available to a Process Designer

ANSWER: B D

Explanation:

A custom BPMN editor stencil is used to tailor the set of BPMN artifacts that a Process Designer can model in Alfresco Process Services. “To make custom process elements available to a Process Designer” is correct because a stencil defines the metadata, visual representation, properties, and editor behavior for a process-model element. This allows an application to expose a purpose-built process artifact in the BPMN designer rather than limiting designers to the standard palette alone.

“To limit the process elements that are available to a Process Designer” is also correct because a customized stencil set can control the palette presented by the editor. Organizations can provide only the supported or approved BPMN elements for a particular modeling context, which helps keep process definitions consistent with the capabilities and conventions of the deployed application. Stencil customization therefore concerns the BPMN process-editor palette and its process-element definitions, including both extending and constraining that palette. Alfresco’s Process Services developer documentation describes application and editor customization concepts at Alfresco Process Services Developer Guide; the Process Services source and extension resources are also available at Alfresco Process Services on GitHub.

QUESTION NO: 2

Whatis the purpose of extending the class AbstractUserDetailsAuthenticationProvidef? Choose 1 answer

A. To implement authentication with a different system other than the default.

B. To add new security checks on top of the default authentication

C. To support synchronization of external user details data

D. To provide a different storage mechanism for the user details data

ANSWER: B

Explanation:

To add new security checks on top of the default authentication is correct because Spring Security’s `AbstractUserDetailsAuthenticationProvider` is specifically intended as a base class for username-and-password-style authentication providers that work with `UserDetails`. Its authentication flow loads the user’s details and then delegates credential and account-status validation to subclass extension points. In particular, subclasses implement the user-retrieval behavior and can implement additional authentication checks through `additionalAuthenticationChecks`. This makes the class appropriate when APS needs to retain the established UserDetails-based authentication process while applying extra validation, such as checks against custom credentials, tenant-specific constraints, or other business security requirements before authentication succeeds. The provider also supports the normal authenticated-token creation flow once those checks pass, allowing the custom logic to integrate cleanly with Spring Security’s authentication architecture. The Spring Security API documents this class as an abstract `AuthenticationProvider` for implementations that authenticate against `UserDetails`, and identifies the subclass hooks used for the additional checks. See the AbstractUserDetailsAuthenticationProvider API documentation and Spring Security’s DAO authentication provider reference.

QUESTION NO: 3

A Service Task calls an external system that can occasionally be unavailable. The process must not keep the user-request transaction open while the call is retried. Which design choices support this requirement? Choose 2 answers.

A. Configure an asynchronous continuation for the Service Task.

B. Make the external operation idempotent.

C. Attach a Boundary Error Event to create automatic technical retries.

D. Replace the Service Task with a Script Task.

E. Store the request state in mutable fields of a singleton Spring Bean.

ANSWER: A B

Explanation:

Configuring an asynchronous continuation causes the engine to persist execution and resume the work through the job executor in a separate transaction. The delegate must also be idempotent, because a failed asynchronous job can be retried and the external call may be attempted more than once.

A Boundary Error Event is intended for modeled BPMN business errors; it does not itself create asynchronous retry behaviour for technical failures. A Script Task still runs synchronously unless an asynchronous continuation is configured. Storing execution-specific state in a singleton bean does not provide retry safety and can introduce concurrency problems.

QUESTION NO: 4

A long-running User Task must be escalated after two days without preventing the original assignee from completing it before the deadline. Which BPMN modelling choices support this requirement? Choose 2 answers.

A. Attach a non-interrupting Boundary Timer Event to the User Task.

B. Create the supervisor escalation work from the timer event's outgoing flow.

C. Attach an interrupting Boundary Timer Event to the User Task.

D. Place an Intermediate Timer Catch Event after the User Task.

E. Attach a Boundary Error Event to the User Task.

ANSWER: A B

Explanation:

A non-interrupting Boundary Timer Event attached to the User Task creates a parallel escalation path when the deadline is reached while leaving the original User Task active. The escalation path can create a separate notification or escalation task for a supervisor.

An interrupting Boundary Timer Event would cancel the attached User Task when it fires, which conflicts with the requirement. An Intermediate Timer Catch Event placed after the User Task only starts waiting after the task is completed. A Boundary Error Event reacts to a BPMN error rather than elapsed time.

QUESTION NO: 5

Which syntax is valid to format a document template variable to upper case? Choose 1 answer

A. < < [myvariable] toString().toUpperCase() > >

B. < < [variables get("myVariable").toUpperCase()] > >

C. < < (variables get("myVariable").toString().toUpperCase()] > >

D. < < [myvariable.toUpperCase()] > >

ANSWER: A

Explanation:

`<< [myvariable] toString().toUpperCase() >>` is valid APS document-template syntax for rendering a variable in uppercase. APS document templates use a placeholder enclosed by << and >>, with the process-variable name placed in square brackets. A Java-style method expression can follow that variable reference. Calling `toString()` first obtains the variable's string representation, and `toUpperCase()` then converts its characters to uppercase before the value is inserted into the generated document. This is especially useful when a variable might not have been declared specifically as a string but is expected to produce textual output in the template. The closing parentheses are essential: they complete both method invocations before the template placeholder closes. This follows the APS document-template approach of resolving process variables and applying supported expression methods during document generation. See the Alfresco Process Services documentation for APS usage guidance and the Java String.toUpperCase() API reference for the behavior of the uppercase conversion.

QUESTION NO: 6

An APS process sends a request to an external partner and must wait for the response that belongs to that specific request. The response must not wake unrelated process instances. Which BPMN event should be used to model the wait for the response? Choose 1 answer.

A. An Intermediate Message Catch Event

B. An Intermediate Signal Catch Event

C. An Intermediate Timer Catch Event

D. An Intermediate Error Catch Event

ANSWER: A

Explanation:

An **Intermediate Message Catch Event** is appropriate for receiving a directed communication from an external participant. The message can be correlated with the intended waiting execution, so a response is delivered to the relevant process instance rather than broadcast to every listener.

A Signal Event has broadcast semantics and can notify multiple matching catches. Timer Events wait for time-based conditions, while Error Events model BPMN error propagation rather than an external business response.

QUESTION NO: 7

A custom JavaDelegate is exposed through a Delegate Expression and is resolved to a singleton Spring Bean. The delegate is used concurrently by many process instances. Which implementation practices are appropriate? Choose 2 answers.

A. Keep execution-specific values in local variables inside execute().

B. Read process data from the DelegateExecution supplied to execute().

C. Store the current process instance identifier in a mutable instance field.

D. Use field injection to hold values that vary by execution.

E. Create a new ProcessEngine for each delegate invocation.

ANSWER: A B

Explanation:

Execution-specific values should be kept in local variables inside the `execute` method, and process data should be obtained from the supplied `DelegateExecution`. These practices avoid sharing mutable state between concurrent invocations of the singleton bean.

Using mutable instance fields for execution-specific values is unsafe because the same bean can serve multiple executions at the same time. Field injection on a reusable delegate expression can also create thread-safety concerns when injected values are mutable. Creating a new ProcessEngine from the delegate bypasses the managed APS engine configuration and is not an appropriate solution.

QUESTION NO: 8

When integrating Alfresco Process Services with an external system, what should the getPassword{) method of the implemented ExternalIdmUser interface return? Choose 1 answer.

A. The password encrypted with SHA-128

B. The unencrypted password

C. The password encrypted with SHA-256

D. NULL

ANSWER: C

Explanation:

The password encrypted with SHA-256 is correct. In an Alfresco Process Services external identity-management integration, an implementation of `ExternalIdmUser` supplies user data to APS during authentication. The value returned by `getPassword()` must be compatible with APS's password-comparison mechanism rather than exposing a clear-text credential. APS expects the password value as a SHA-256 hash, allowing the authentication layer to compare the submitted password after applying the corresponding hashing process. This is important both for interoperability with the APS identity subsystem and for safe credential handling: the external user object should not provide a plain-text password to the application. The external IDM extension point is intended to let APS obtain users and groups from a corporate directory, custom database, or another identity source while maintaining the password representation required by its authentication flow. When implementing the integration, ensure that the external system's stored or calculated hash uses the encoding expected by APS and that password data is handled only over appropriately protected connections. See Alfresco's external identity-management guidance in the Alfresco Process Services documentation and the Process Services documentation portal.