DUMPSBOSS.

Pega Certified Robotics System Architect (PCRSA) 80V1 2019

Pegasystems PEGAPCRSA80V1_2019

Version Demo

Total Demo Questions: 9

Total Premium Questions: 94

Buy Premium PDF

https://dumpsboss.co

support@dumpsboss.co

support@dumpsboss.co
dumpsboss.co

Topic Break Down

Topic	No. of Questions
Topic 1, Robotic solution design	11
Topic 2, Robotic solution development	58
Topic 3, Robotic solution testing	18
Topic 4, Robotic solution deployment	7
Total	94

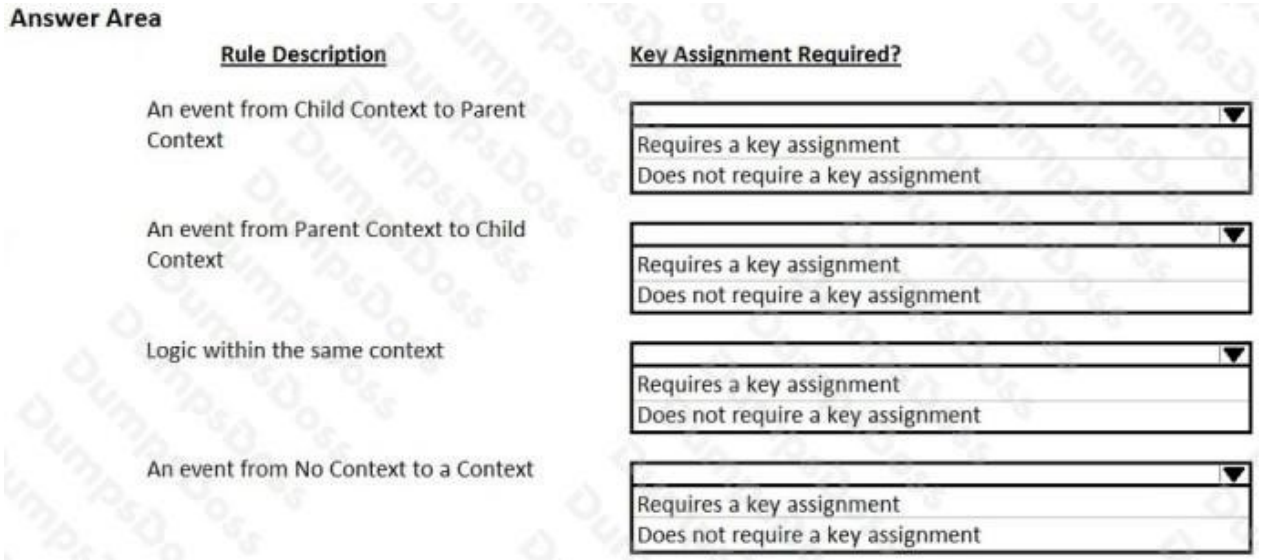
QUESTION NO: 1 - (HOTSPOT)

HOTSPOT

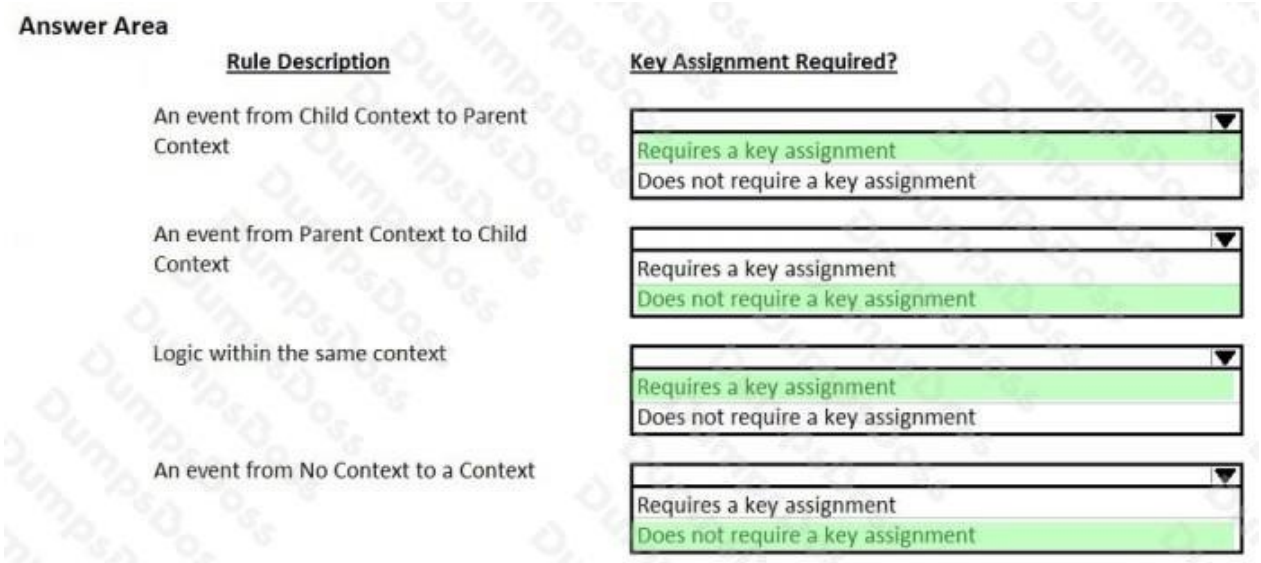
Pega Robot Studio provides five rules on how to differentiate between cloneable application objects when using key assignments in automations. The first rule states that an event creates the instance to set the context of a cloneable object. The remaining four rules state the requirements for a key assignment.

In the Answer Area, determine if each rule description requires a key assignment.

Hot Area:



ANSWER:



Explanation:

Cloneable application objects can represent multiple runtime instances of the same interrogated control, so Robot Studio must know which specific instance an automation event or logic step is intended to use. A key assignment provides that instance-identification information when the automation flow cannot determine the correct clone solely from the context that is already established. The event that creates an instance establishes the initial context; the remaining rules describe when an explicit key is needed as work moves through or across contexts.

For an event moving from Child Context to Parent Context, a key assignment is required because the parent-level action needs to resolve the relevant cloned instance from information originating below that parent context. Logic that remains within the same context also requires a key assignment, since several clones may be present in that context and the

automation must explicitly identify the intended one. These assignments make the automation deterministic and prevent it from acting on an unintended runtime instance.

An event moving from Parent Context to Child Context does not require a key assignment because the already-established parent context supplies the information needed to scope the child instance. Likewise, an event moving from No Context to a Context does not require a key assignment because that event establishes the context for the cloneable object. This reflects the fundamental principle that context creation supplies the starting instance identity, while actions needing to distinguish among existing instances require a key. See the Pega Academy Robot Studio learning resources and the Pega documentation portal for Robot Studio automation and application-object guidance.

QUESTION NO: 2

During project testing, an issue requires you to add a diagnostic log component to track the log files to help determine a resolution. After testing, you decide not to remove the diagnostic log component from the automation and decide to simply turn off the log component.

Which diagnostic log component setting allows you to turn the logging component off temporarily?

A. Setting the Category to Off

B. Setting the Mode to Off

C. Setting Type to Off

ANSWER: B

Explanation:

Setting the Mode to Off is correct because the diagnostic logging component's Mode setting controls whether that configured component actively writes diagnostic events at runtime. Selecting Off disables logging for the component without deleting the component or its configuration from the automation. This is useful after troubleshooting: the diagnostic setup remains available for later investigation, while routine execution avoids producing unnecessary log output and consuming associated disk space. When a related issue must be investigated again, the component can be re-enabled by changing its mode to the desired logging behavior, preserving the existing diagnostic configuration and avoiding the need to recreate it. In Pega Robot Runtime configuration, diagnostic logging is designed to be configurable so that logging can be enabled during analysis and suppressed when it is no longer needed. The Mode property is the operational switch for this temporary state. See the Pega Robot documentation landing page at Pega Robotic Automation documentation and the legacy RuntimeConfig reference at RuntimeConfig.xml reference.

QUESTION NO: 3

You are designing an automation that adds new customers to an online insurance web site. If a customer exists, a pop-up window is displayed with an error message, which closes automatically after 35 seconds.

While debugging the automation, you notice that you are receiving a control not created exception message in the Add Customer procedure after a page navigation occurs.

How do you resolve the exception?

A. Add a missing created event in the Add Customer procedure.

B. Ensure the waitForCreate timeout is longer than 35 seconds in the Add Customer procedure.

C. Ensure the isCreated timeout is longer than 35 seconds in the Add Customer procedure.

D. Add a missing waitForEvent.Exists method in the Add Customer procedure.

ANSWER: D

Explanation:

Add a missing waitForEvent.Exists method in the Add Customer procedure. A page navigation causes web controls from the prior document state to be destroyed and then recreated as the new page is loaded. An automation must therefore synchronize with the target control's lifecycle before attempting to read from it, click it, or invoke another control method. Waiting for the `Exists` event explicitly pauses the procedure until Robot Studio detects that the control is available in the newly loaded page context. This prevents the procedure from continuing while the control has not yet been created, which is the condition that produces the control-not-created exception.

This synchronization is especially important when the application can display an existing-customer message in a transient pop-up and then automatically close it. The automation should wait for the relevant post-navigation control state rather than assume a fixed rendering time. Event-based waiting makes the procedure responsive to the actual application state and avoids timing-dependent behavior across different machines or network conditions. Pega's robotic-automation learning resources and product documentation provide the core guidance for designing automations around application events and control availability: Pega Academy and Pega Documentation.

QUESTION NO: 4

You are designing a procedure that receives a customer identifier and returns a status message to the calling automation.

Which two practices support a reusable procedure design? (Choose two.)

A. Define a typed input parameter for the customer identifier.

B. Return a status message through an output or exit path.

C. Hard-code the customer identifier in the procedure.

D. Use an application event as the procedure's only entry mechanism.

ANSWER: A B

Explanation:

Defining a typed input parameter for the customer identifier creates a clear interface for callers and ensures that the procedure receives compatible data. Returning a status message through an output or exit path gives each caller a consistent way to handle successful completion or a business-processing issue.

Hard-coding a customer identifier prevents reuse, and connecting an event handler directly as the procedure entry mechanism does not provide the standard callable procedure interface. See Pega documentation on automation types.

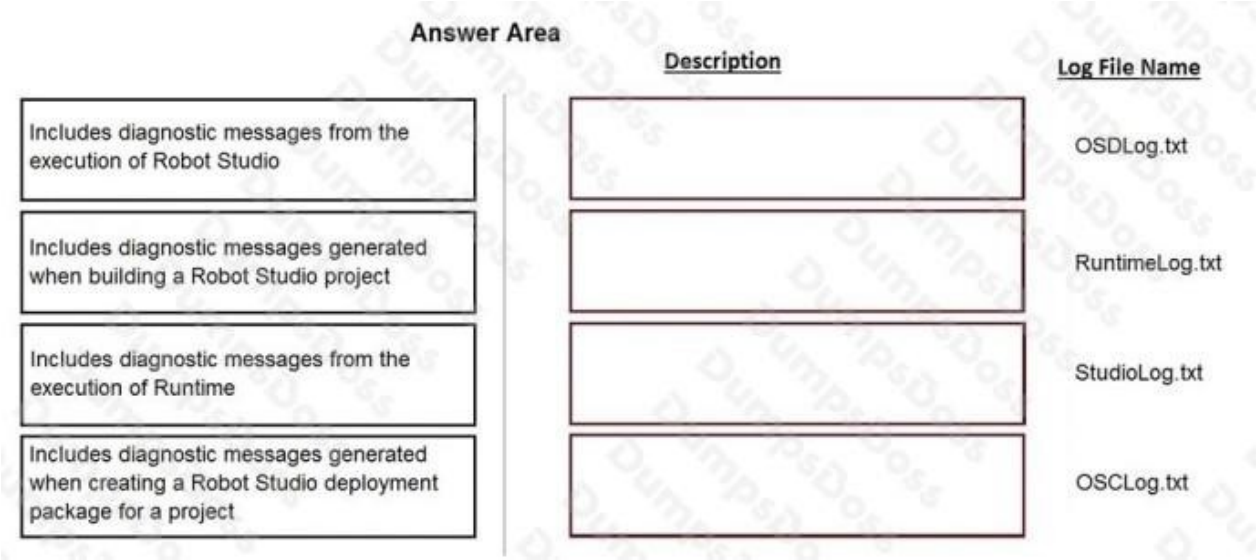
QUESTION NO: 5 - (DRAG DROP)

DRAG DROP

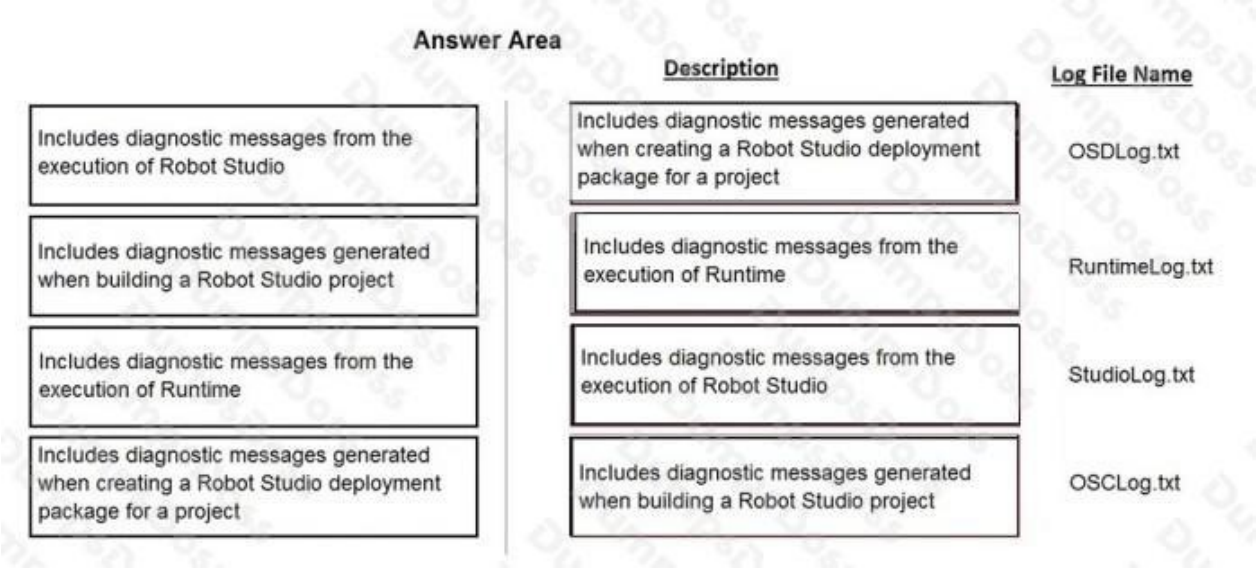
Pega Robot Studio produces several log files during the opening, building, debugging, and deployment of a solution.

In the Answer Area, drag the description of the log file on the left to its correct log file name.

Select and Place:



ANSWER:



Explanation:

Pega Robot Studio writes separate diagnostic logs for distinct stages and hosts involved in a robotic solution's lifecycle. The correct mapping preserves that separation. **OSDLog.txt** is the log associated with creation of a Robot Studio deployment package, so it contains diagnostic messages generated while the deployment package is being created. **RuntimeLog.txt** is produced by the Runtime environment; therefore, it records diagnostics from execution of Runtime. **StudioLog.txt** is the Robot Studio execution log and contains diagnostics generated while Robot Studio itself is executing. **OSCLog.txt** is associated with the project compilation/build process, so it records diagnostic messages generated when building a Robot Studio project.

This organization lets an automation developer isolate troubleshooting to the relevant part of the workflow: package-generation activity is reviewed in the deployment log, a running bot is investigated through the Runtime log, Studio-host activity is investigated through the Studio log, and build activity is investigated through the compilation/build log. Logging is configured and used as a diagnostic aid in Robot Studio, as described in the Pega documentation's logging configuration guidance. The mapping shown in the answered image correctly associates each filename with the stage that generates its messages, providing an appropriately targeted record for debugging, building, running, and packaging a solution.

QUESTION NO: 6 - (SIMULATION)

In the Answer Area, drag each use case on the left to the correct Toolbox item.

ANSWER: See the explanation for the answer

Explanation:

The provided JSON does not include the Answer Area image or the labels for either the use cases or the Toolbox items. Therefore, the required drag-and-drop mappings cannot be determined from the supplied question text.

Please provide the screenshot (or list the use cases and Toolbox item names) to identify the exact correct matches.

QUESTION NO: 7

You are configuring a project for deployment to a test environment and a production environment.

Which two values are appropriate to place in Project Configuration files? (Choose two.)

A. The target application URL

B. A shared-folder path used for exported files

C. The hierarchy of interrogated web controls

D. The execution links between automation components

ANSWER: A B

Explanation:

Environment-specific application URLs and file-system paths are appropriate configuration values because they commonly differ between test and production while the underlying automation remains unchanged. Externalizing these values allows one solution to be deployed with the configuration that applies to the selected environment.

Interrogated controls and automation flow logic are project design assets and should not be duplicated in configuration files to support environment changes. See Pega documentation on Project Configuration files.

QUESTION NO: 8

An automation selects a value in a web drop-down list. The application enables the Save button only when the list control raises its onchange event.

How do you enable the Save button through the automation?

A. On the drop-down list, add a RaiseEvent method with onchange.

B. On the Save button, add a RaiseEvent method with onchange.

C. On the Save button, add an Enabled property and set it to True.

D. On the drop-down list, add a Disabled property and set it to False.

ANSWER: A

Explanation:

Add a **RaiseEvent** method with `onchange` to the drop-down list control. Setting or selecting a value does not always cause the target application's client-side event handlers to run. Raising the event reproduces the interaction that the web page expects after a user changes the selection.

Raising the event on the Save button does not notify the page that the list value changed. Directly changing the Enabled property is not an appropriate solution because the application controls that state through its own validation logic. See Pega documentation search for RaiseEvent for related product guidance.

QUESTION NO: 9

You are debugging a script component for your project. When the automation executes your script, the intended result does not occur. You suspect that one or more variables within your script are not being set to the proper value. Your version of Pega Robot Studio does not support the use of breakpoints within the script designer.

In what two ways can you debug the script? (Choose two.)

A. Enable Runtime logging and enter a diagnostic message to log the variable values during script execution.

B. Add a breakpoint before calling the script so that you can assign a watch for the variable(s) used within the script.

C. Check with the Support team to upgrade Robotic Automation Studio to the version which contains script debugging.

D. Modify the script so that the variable values are returned in the automation.

ANSWER: A D

Explanation:

Enable Runtime logging and enter a diagnostic message to log the variable values during script execution. Runtime logging provides a practical way to observe the state of script variables while the robot is running. By writing appropriately scoped diagnostic messages from the script, a developer can record the values that are actually calculated or received at key points in execution. Reviewing the runtime log then helps identify where a value first differs from the expected value, without requiring an interactive script breakpoint.

Modify the script so that the variable values are returned in the automation. Exposing relevant values through script outputs lets the calling automation inspect, display, or log those values using normal Robot Studio debugging facilities. This is especially useful when the script designer cannot stop on a line and expose local variables directly. Returning only the values needed for diagnosis keeps the interface understandable and enables the automation to validate intermediate state during a test run. These techniques use supported runtime observability and automation-level data flow to diagnose script behavior. See Pega documentation for runtime logging and the Script component reference.