PassQueen
a.k.a dumpsqueen.com

Certified Pega Robotics System Architect 22

Pegasystems PEGACPRSA22V1

Version Demo

Total Demo Questions: 12

Total Premium Questions: 128

Buy Premium PDF

https://passqueen.com

support@passqueen.com

Topic Break Down

Topic	No. of Questions
Topic 1, Pega Robotics architecture	7
Topic 2, Robotic solution design	11
Topic 3, Robotic solution development	68
Topic 4, Robotic solution debugging	27
Topic 5, Robotic solution deployment	14
Topic 6, Mix Questions	1
Total	128

QUESTION NO: 1

An automation retrieves a customer number from a text control and uses the value as input to a search method on a second application. The search method must not run until the customer number is available.

Which connection type should the developer use to pass the customer number between the two components?

A. A blue data link.

B. A yellow execution link.

C. A Label and Jump To connection.

D. A breakpoint connection.

ANSWER: A

Explanation:

A **blue data link** passes a value from an output property to a compatible input property. In this scenario, the customer number produced by the first control is required by the search method, so the value must be connected by a data link.

Yellow execution links control the sequence in which automation steps run, but do not transfer values. The developer can use both link types: an execution link to sequence the work and a blue data link to provide the customer number to the search method.

QUESTION NO: 2

A project requirement is to run the solution in multiple environments: Development and Production.

Which two items can be added to the two Project Configuration files? (Choose two.)

A. Variable values

B. Citrix Context properties

C. Adapter Text MatchRules

D. Project properties

ANSWER: A D

Explanation:

"Variable values" and "Project properties" are the configuration items used to make a Pega Robot Studio solution portable between environments such as Development and Production. A project configuration file provides environment-specific values without requiring developers to alter the automation design itself. For example, a variable can hold a value that differs by environment, such as a service endpoint, a folder location, or an application-specific setting. When the solution is prepared for another environment, the relevant configuration file supplies the appropriate value.

Project properties can likewise be placed in the configuration file so that project-level settings are selected according to the target environment. This separation of configuration from implementation supports controlled deployment, minimizes manual changes after packaging, and helps ensure that the same solution artifacts can be promoted consistently through the delivery lifecycle. Pega Robotics configuration and deployment practices are designed around maintaining this distinction between reusable automation logic and environment-dependent settings. See the Pega Robotic Automation product documentation entry point and the Pega Academy for Pega Robotics learning and configuration guidance.

QUESTION NO: 3

A developer adds a constant named **const_DefaultCountry** in the Globals designer. Several automations in the same robotic project must use the value when creating customer records.

What is the scope of the constant?

A. It is available to every automation in the project.

B. It is available only to the first automation that references it.

C. It is available only to automations in the same application project item.

D. It is available only while the Globals designer is open.

ANSWER: A

Explanation:

A variable or constant added in the Globals designer has project-level scope. Therefore, every automation in the project can reference `const_DefaultCountry`.

The `const_` prefix indicates that the value is intended to remain constant; it does not make the item local. A locally created item is available only within the automation in which it is defined.

QUESTION NO: 4 - (SIMULATION)

Match the term on the left with its description on the right.

ANSWER: See the explanation for the answer

Explanation:

Correct matches:

Data pages — Source application data from Pega Platform applications by calling automations.
Flow action — Within a case, calls automations to run before or after completing a step.
Robot queue — One or more robots access the case assignment to perform the automations.

Data pages support robotic data lookup/enrichment, flow actions invoke desktop automation as part of an interactive case step, and robot queues deliver assignments to unattended robots for processing.

QUESTION NO: 5

A solution has two automations that run on separate execution threads. The second automation occasionally attempts to use a value before the first automation has finished producing it.

Which two debugging actions provide the most useful evidence for investigating the execution order? *(Choose Two)*

A. Add studio execution log entries at the producing and consuming steps.

B. Use Automation Playback to review the execution path.

C. Add Try and Catch components to both automations.

D. Replace the data link with an execution link.

E. Remove all breakpoints before investigating the issue.

ANSWER: A B

Explanation:

Adding **studio execution log entries** creates timestamped checkpoints that can show when the producing automation sets the value and when the dependent automation attempts to consume it. **Automation Playback** provides a visual view of the execution path that can be correlated with the log entries.

Together, these tools help establish whether the issue is caused by timing, sequencing, or a missing value. Adding exception handling might prevent an unhandled error, but it does not establish the order in which the two threads executed.

QUESTION NO: 6

Which two of the following Tool Windows are used in Pega Robot Studio? (Choose two.)

A. Designer windows

B. Solution Explorer

C. Object Explorer

D. Menu toolbar

ANSWER: B C

Explanation:

Solution Explorer and **Object Explorer** are Tool Windows used in Pega Robot Studio. Solution Explorer provides the project-level view of a Robot Studio solution, including its projects and the files or configuration elements that belong to them. This lets developers organize, select, and manage the assets used to build robotic automations. Object Explorer is the development view used to inspect the objects in an automation project, such as adapters and matched controls, and to work with their available properties, methods, and events. These windows support the core Robot Studio workflow: Solution Explorer manages the solution structure, while Object Explorer exposes the application objects that automations interact with. Pega Robot Studio is built on the Visual Studio isolated shell, so its interface includes dockable tool windows that provide specialized development views alongside the designers. Pega documentation describes the Robot Studio development environment and its project/object-oriented tooling, while Microsoft documents the general Visual Studio tool-window model that underlies this style of interface. See Pega Robot Studio documentation and Visual Studio window documentation.

QUESTION NO: 7

Which two statements are valid for the given automation? (Choose two.)

A. The automation is initiated when the value of the NearestStore changes.

B. Assign Main|b|NearestStore with the new value only if the value changed belongs to an active key.

C. Assign Main|b|NearestStore with the new value when the value of the NearestStore changes.

D. The value of the NearestStore changes when the automation is initiated.

ANSWER: B D

Explanation:

The automation's configuration updates `Main|b|NearestStore` as part of its execution, so "The value of the NearestStore changes when the automation is initiated." accurately describes the sequence: initiating the automation causes the assignment that updates the data value. The assignment is also scoped to an active key. Therefore, "Assign Main|b|NearestStore with the new value only if the value changed belongs to an active key." is valid because keyed data handling ensures that a value update is applied only in the currently active keyed context. This prevents a value intended for one keyed instance from being written to another instance of the same data structure. In Pega Robot Studio, automations can use data values and keyed data to maintain context while interacting with applications, and the automation flow determines whether an event initiates an automation or an automation performs a value update. Here, the flow is an initiated automation that performs the update, with the active-key condition governing the target context. See the Pega Academy for

Robot Studio learning resources and the Pega Documentation portal for product documentation on robotic automation and data handling.

QUESTION NO: 8

A robotic solution is promoted from Development to Production. The automation uses a variable for an application service endpoint, and the endpoint value differs in each environment.

Where should the developer place the environment-specific endpoint value so that the automation design does not need to change?

A. In a variable value in the Project Configuration file.

B. In the Element Path match rule of the application control.

C. In a local variable within every calling automation.

D. In the PegaRuntimeConfig.xml startup-project setting.

ANSWER: A

Explanation:

The endpoint should be stored as a **variable value in the Project Configuration file**. Project Configuration files allow a solution to use different environment-specific values while preserving the same automation implementation.

Changing an interrogated control match rule or editing automation logic for each environment makes deployment more difficult and risks introducing inconsistent behavior. Project properties and variable values are intended to separate environment configuration from solution design.

QUESTION NO: 9

To modify an object's default properties before use in a project, which setting must be updated?

A. Naming Rules

B. Prefix Types

C. Type Prefixes

D. Type Name

ANSWER: A

Explanation:

Naming Rules is correct because Pega Robot Studio uses naming rules to define the default naming and related property conventions applied when objects are created or interrogated for use in a project. Updating these rules before creating the project objects ensures that newly generated objects receive names and default property values that align with the organization's implementation standards. This is useful for making automations easier to read, maintaining consistent object identification, and reducing manual cleanup after interrogation.

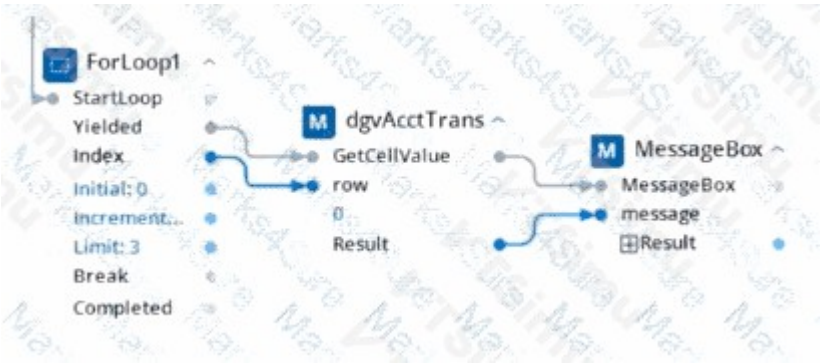
These settings are intended to be configured at the environment or Studio level before the developer begins adding application objects, controls, and automation components. Once the naming rules are established, Robot Studio can apply the configured conventions consistently as it creates project artifacts. This supports maintainability because team members encounter predictable names and object metadata across the solution. Pega's Robot Studio documentation describes the configuration capabilities available in the Studio environment, while Pega Academy's robotics learning material covers the design practices used when building robotic automations. See the Pega Robot Studio documentation and the Pega Academy robotic automation topic.

QUESTION NO: 10

in the BankerInsight application, a gdvAcctTrans control provides the history of a customer ' s account transactions, as shown in the following figure:



Consider the following automation, where the Increment property of the For Loop component equals 1:



What is the order of the output that is displayed in the Message Box windows?

A. 3255, 1763, 3451.

B. The automation throws an out of bounds exception.

C. - > , NULL, NULL.

D. 3255, 05/05/2014, Cash deposit.

ANSWER: A

Explanation:

3255, 1763, 3451. is correct because the For Loop supplies successive zero-based row indexes to the data-grid cell retrieval: 0, then 1, then 2. Its increment of 1 advances the index by one on each pass, and the loop limit stops processing before index 3. The column input for *GetCellValue* remains set to 0, so every retrieval is from the first data column rather than moving across columns. In the displayed transaction grid, column 0 contains the transaction IDs. Consequently, the retrievals are the value at row 0/column 0 (3255), followed by row 1/column 0 (1763), and row 2/column 0 (3451). Each retrieved value is passed to the Message Box during its respective loop iteration, producing those values in that sequence. This follows the standard zero-based row and column indexing model used by DataGridView-style controls; see the DataGridView API reference. Pega Robotics automations use control methods and loop components to perform this kind of repeated interaction with an interrogated application control; see the Pega Academy for Robotics learning resources.

QUESTION NO: 11 - (SIMULATION)

Before deploying your robotic project, you realize that the connection parameters (or Pega Robot Manager, the Pega Robot Runtime settings, and the application login credentials for Assisted Sign-On need updating to reflect the production environment.

Arrange the three steps, as shown in the following figure. that you click in the correct order to access the necessary configuration files. (Choose Three).

ANSWER: See the explanation for the answer

Explanation:

Correct click sequence:

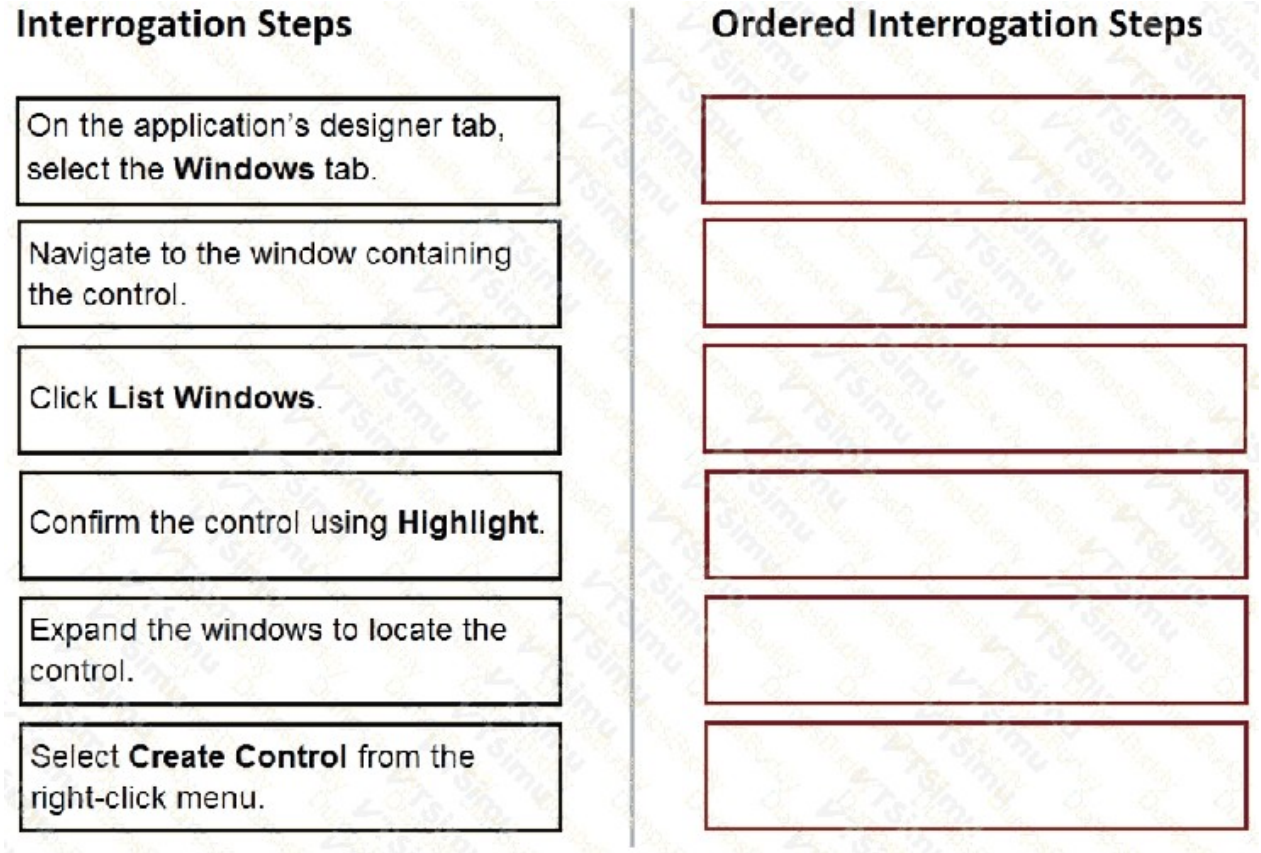
1. **Tools**
2. **Folders**
3. **Files**

In Pega Robot Studio, use the `Debug → Tools → Folders → Files` navigation path to open the folder containing the deployment configuration files. Update the applicable files there—for example, the Robot Manager connection, Pega Robot Runtime configuration, and Assisted Sign-On credential settings—so they use the production values before deployment.

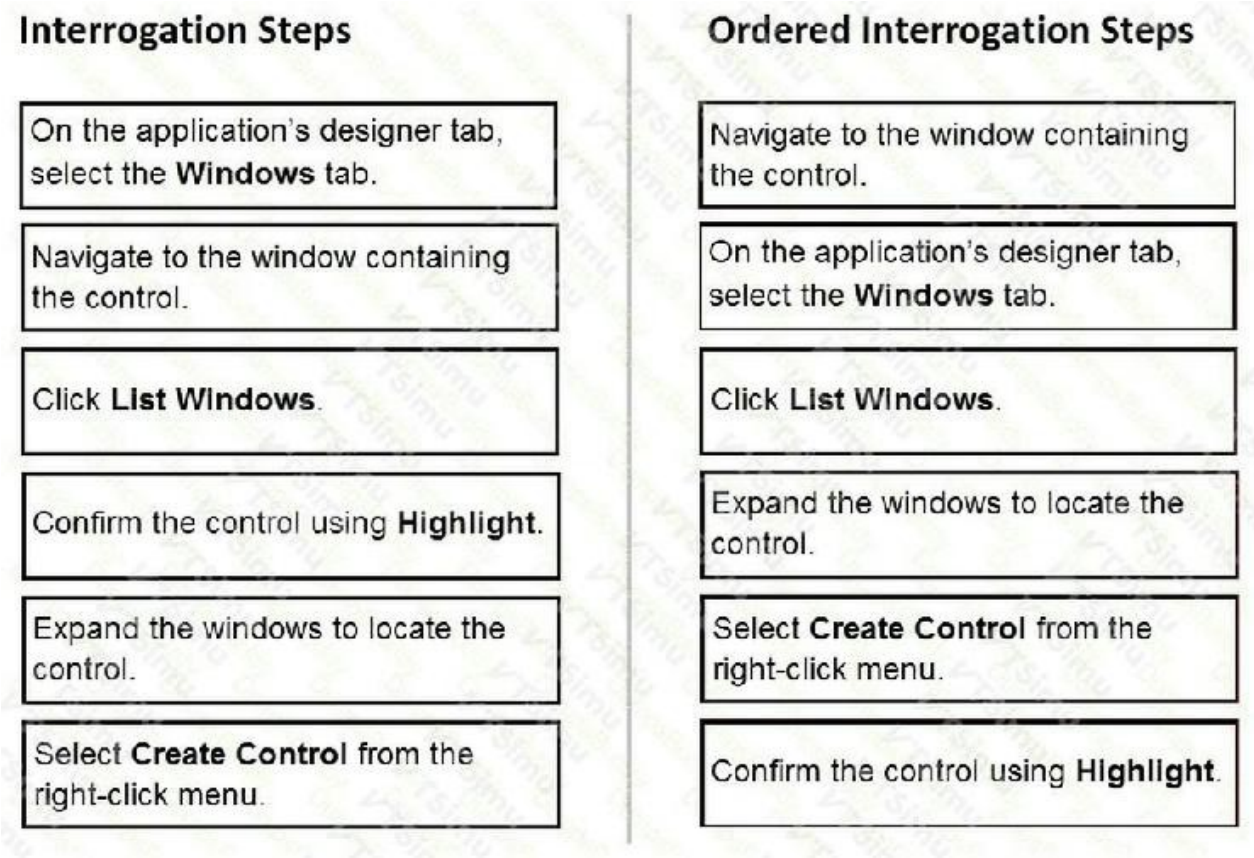
QUESTION NO: 12 - (DRAG DROP)

When interrogating a Windows control, the drag and drop Default interrogation method does not work. You decide to use the Create Control option to interrogate the control.

From the Interrogation Steps list, move all of the options to the Ordered Interrogation Steps column and place them in the correct order.



ANSWER:



Explanation:

Creating a control manually is used when the usual drag-and-drop interrogation approach cannot identify a Windows UI element. The process starts with the target application already positioned on the screen that contains the control. This matters because Pega Robotics needs the relevant application window to exist and be available when its Windows inventory is refreshed.

In Application Designer, the Windows tab is the place where Pega Robotics displays the discovered application windows and their control hierarchy. Selecting **List Windows** refreshes and populates that hierarchy. Once the list is available, expanding the windows is the step that exposes the nested elements until the required control can be found. The manual interrogation action is then initiated by choosing **Create Control** from the target element's context menu. This creates the control definition from the selected native UI element rather than from a dragged interrogation target.

After the control has been created, **Highlight** verifies that the resulting adapter control maps to the intended on-screen element. The highlight action provides a visual check of the relationship between the configured control and the live application UI, which is especially useful for controls selected through a hierarchy instead of directly interrogated. This sequence follows the Windows-based control discovery and verification workflow used by Pega Robot Studio. See the Pega documentation on interrogating applications and the Pega Academy interrogation topic for the related interrogation concepts.