DUMPSBOSS.

# LSA Pega Architecture 74V1

## Pegasystems PEGACLSA74V1-A

### Version Demo

**Total Demo Questions: 8**

**Total Premium Questions: 83**

**Buy Premium PDF**

https://dumpsboss.co

support@dumpsboss.co

support@dumpsboss.co
dumpsboss.co

# Topic Break Down

| Topic | No. of Questions |
|---|---|
| Topic 1, Application Design | 12 |
| Topic 2, Case Management | 9 |
| Topic 3, Data Model | 7 |
| Topic 4, User Interface | 8 |
| Topic 5, Reporting | 7 |
| Topic 6, Integration | 6 |
| Topic 7, Security | 13 |
| Topic 8, Performance | 11 |
| Topic 9, Deployment | 10 |
| Total | 83 |

QUESTION NO: 1

You are using Pega Express. You want to avoid creating unnecessary case level properties and views.

Which three actions do you take to accomplish this goal? (Choose Three.)

**A.** Define case type view using ‘’. Page.property ‘’syntax for the majority of the filed.

**B.** Drag-and-drop a Filed Group when defining a case view.

**C.** Pre-define properties, sections, and relevant records that are applied to work Cover

**D.** Add an embedded Page property to the case type.

**E.** Create a data type that corresponds to each case type.

**ANSWER: A C D**

**Explanation:**

Using “.Page.Property” references for most fields lets the case view read and write values held on an embedded page rather than requiring a separate top-level property for every field. This keeps the case data model focused on the information that genuinely belongs directly to the case and makes related data easier to organize and reuse.

Predefining properties, sections, and relevant records on the work cover supports reuse across the cases in a case hierarchy. Shared artifacts can be defined once at the cover level and applied where needed, reducing the need to create duplicate case-specific views and properties.

Adding an embedded Page property creates a structured container for related data. The page can represent a logical business entity and its fields can be referenced from case views through page-property notation. Consequently, the case type does not need an extensive collection of unrelated scalar properties at its top level. This approach is consistent with Pega’s data-modeling guidance to use embedded data structures for grouped, reusable business information and to design case data deliberately in App Studio. See Pega’s data-modeling documentation and its Data Model learning topic.

**QUESTION NO: 2**

A Pega application has cases that represent customers’ accounts each with many members. When a members of a customer accounts registers with the application through an offline component, a related, registration transaction is recorded. Am advanced agent updates the customer account cases with new members. The application is running in a multimode system and advanced agents are enabled on all nodes.

Which two element are valid design choices (Choose two.)

**A.** Use the optimistic locking option on the case types.

**B.** Create a Registration subcase configured to run in offline mode.

**C.** Leverage the default object lock contention queuing capability.

**D.** Override DetermineLockString to use Account instead of .PyID as the lock string

**ANSWER: B C**

**Explanation:**

Creating a Registration subcase configured to run in offline mode is a sound separation of responsibilities. The registration is an independently recorded business event that can be captured while connectivity is unavailable and subsequently synchronized. Modeling that event as its own case preserves its lifecycle and audit trail, while allowing the account case to be updated later by background processing. This avoids requiring the offline interaction to directly perform a potentially contended update to the customer account.

Leveraging the default object lock contention queuing capability is also appropriate because multiple advanced-agent requestors, potentially executing on different nodes, can attempt to update the same account case at nearly the same time.

Pega's lock handling can defer processing that cannot immediately obtain the required object lock and retry it later. This serializes conflicting account updates without losing a registration event and supports reliable processing in a multinode deployment. The agent should perform the account update as a short, committed transaction after obtaining the lock. Pega's general product guidance and training material on offline processing and background processing are available through Pega Documentation and Pega Academy.

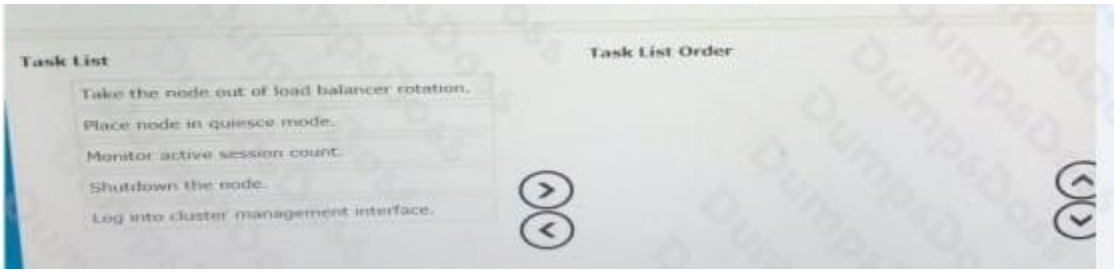
## QUESTION NO: 3 - (DRAG DROP)

DRAG DROP

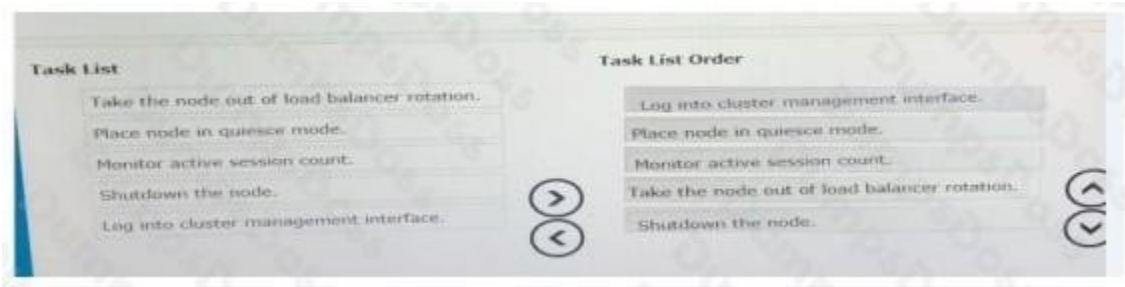
BigCo is planning an update to JVM settings on all production nodes at the of the month.

The production application supports high availability.

To ensure that users can continue working without distribution while the JVM changes are in process, select and move all of the following tasks to the Task List Order column arrange then correct order.




## ANSWER:




**Explanation:**

For a rolling JVM-settings update on a highly available Pega production system, each node must be drained safely before it is shut down. Start by logging in to the cluster management interface so the node's operating state can be controlled and observed through the supported cluster-management mechanism. Next, place the node in quiesce mode. Quiescing is the key first transition because it prevents the node from accepting new work while allowing work that is already active to finish normally. This protects users from interruption and avoids terminating work in progress.

After the node is quiesced, monitor the active session count until the active work has drained to the required level, normally zero for planned maintenance. Once the node no longer has active sessions to serve, take it out of load balancer rotation. At that point, traffic is no longer routed to the node and the node has been drained in an orderly way. The node can then be shut down to apply the JVM configuration change. Repeat this controlled sequence one node at a time so other nodes continue handling user requests throughout the maintenance window. Pega's high-availability approach depends on keeping sufficient healthy nodes available while an individual node is removed for maintenance. See Pega Platform node management documentation and Pega guidance for maintaining system availability.

## QUESTION NO: 4

A benefits application uses a standard eligibility decision table. A new regulation changes the eligibility thresholds beginning on January 1 of the next calendar year.

The current and future threshold logic must coexist, and the application must select the correct logic according to the case effective date.

How do you configure the application?

**A.** Create a date-circumstanced version of the eligibility decision table.

**B.** Withdraw the current decision table and replace each threshold with the future value.

**C.** Create a new ruleset version and copy the decision table without circumstancing.

**D.** Configure a Declare Trigger rule to update the thresholds when the case is opened.

## ANSWER: A

**Explanation:**

Create a date-circumstanced version of the eligibility decision table. Date circumstancing is appropriate when a rule variation becomes effective on a known date. Pega can continue to resolve the base rule for cases before the effective date and select the circumstanced version when the case date meets the circumstance condition.

This approach keeps the existing rule available for historical cases and avoids procedural logic that selects between separate rule names. It also provides a clear audit trail of the rule change required by the regulation. See Pega Academy: Circumstancing.

## QUESTION NO: 5

You imported a Rule- Ad mm- Product (RAP) file that contains new features into a user acceptance testing (UAT) environment.

Select the option that ensures basic functionality behaves as expected before inviting users to start acceptance testing.

**A.** Create a smoke test suite. Run the test suite using the pre-import process.

**B.** Create a regression test suite. Run the test suite using the automation server.

**C.** Create an integration test suite. Run the test suite using the Execute Test REST service.

**D.** Create an ad hoc test suite. Run the test suite using the post-import process.

**E.** Create a smoke test suite. Run the test suite using the post-import process.

## ANSWER: E

**Explanation:**

Create a smoke test suite. Run the test suite using the post-import process. Smoke testing provides a fast, focused validation that the application's essential paths and newly deployed rules are operational after the RAP import. This is the appropriate confidence check before business users begin UAT: it confirms that the deployed application starts correctly and that the critical functionality required to conduct acceptance testing is available.

The timing is essential. A post-import process executes after the RAP archive has been imported into the UAT environment, so the smoke suite evaluates the actual rule set and configuration that users will test. A pre-import execution would only validate the environment before the new features are present and therefore cannot establish that the imported features have deployed successfully. Pega test suites support organizing automated test cases for targeted execution, including lightweight checks used to validate a deployment. See Pega's testing applications documentation and the Pega Academy topic on testing applications.

## QUESTION NO: 6

An end user of the application experienced a browser crash working on a highly available system. Crash recovery is enabled.

Does the user have to be authenticated?

**A.** Yes, if the user is directed to the same node.

**B.** Yes. If the authentication cookie was lost.

**C.** No, if the authentication cookie was lost.

**D.** No, if the user is directed to a different node.

**ANSWER: B**

**Explanation:**

**Yes. If the authentication cookie was lost.** Crash recovery enables Pega to restore a user's interrupted work after a browser failure, including in a highly available environment where the replacement request can be served by an appropriate node. Recovery of the application context, however, does not replace the browser's proof of an authenticated identity. Pega relies on the authentication mechanism and its browser-held session or authentication cookie to associate an incoming request with the previously authenticated operator.

If that cookie remains available after the browser restarts, the user can normally be recognized and the recoverable session context can be resumed. If the authentication cookie has been lost, the new browser request cannot establish that it belongs to the authenticated user. The user must therefore authenticate again before Pega can grant access and continue recovery. This preserves the security boundary between restoring work state and proving the identity of the person requesting that state. Pega's platform documentation describes authentication as the process used to identify users and establish their access, while high-availability configuration supports continued processing when infrastructure or request handling changes. See Pega Platform authentication documentation and Pega Platform node classification documentation.

**QUESTION NO: 7**

Which three approaches are considered a PegaUnit testing best practice? (Choose Three)

**A.** A test case uses limited assertions.

**B.** A test ruleset is placed at the top of an application's ruleset stack.

**C.** A test case focuses on a single area of functionality.

**D.** A test case groups relevant assertions together .

**E.** A test case is not necessary when un error message is the expected result.

**ANSWER: A C D**

**Explanation:**

"A test case uses limited assertions," "A test case focuses on a single area of functionality," and "A test case groups relevant assertions together" reflect the core design principles for maintainable PegaUnit tests. A unit test should have a narrow, understandable purpose so that a failure quickly identifies the functional behavior that needs attention. Keeping the scope focused on one area of functionality also helps prevent unrelated rule changes from making a test difficult to diagnose or maintain.

Assertions should be deliberately limited to the outcomes that establish the behavior under test. This keeps the test concise while avoiding unnecessary coupling to incidental implementation details. At the same time, assertions that validate related aspects of the same scenario belong together. For example, a test of a validation outcome can assert the relevant status and message results in one focused case, provided both assertions are part of that single behavior. These practices produce tests that are readable, stable, and useful as regression coverage throughout application development. See Pega documentation on unit testing with PegaUnit and the Pega Academy material on unit testing.

## QUESTION NO: 8

A case worker updates an address in a customer case. The application must automatically update a calculated property that determines whether the customer is eligible for a regional service.

The calculation must be reevaluated whenever any of the address properties used by the calculation changes.

Which rule type do you use?

**A.** Declare Expression

**B.** Declare Trigger

**C.** Decision Tree

**D.** Validate rule

## ANSWER: A

**Explanation:**

Use a Declare Expression rule. A Declare Expression defines a target property and the source properties on which that target depends. When a source value changes, Pega automatically recalculates the target property, keeping the eligibility value synchronized with the address data.

This declarative approach is preferable to invoking an activity from every UI or flow location that might update the address. It centralizes the calculation and ensures that the dependency is maintained consistently. See Pega documentation on Declare Expression rules.