DUMPSARENA

Certified Senior System Architect (CSSA) 72V1

Pegasystems PEGACSSA72V1

Version Demo

Total Demo Questions: 11

Total Premium Questions: 113

Buy Premium PDF

https://dumpsarena.co

sales@dumpsarena.co

sales@dumpsarena.co
dumpsarena.co

Topic Break Down

Topic	No. of Questions
Topic 1, Application Design	32
Topic 2, Case Management	29
Topic 3, Data Modeling	18
Topic 4, User Interface	7
Topic 5, Data Integration	13
Topic 6, Reporting	6
Topic 7, Security	7
Topic 8, Mix Questions	1
Total	113

QUESTION NO: 1

What two actions must you perform to create a class join in a report definition? (Choose Two)

A. Add a parameter for each property in the class you want to join.

B. Select the type of match for key values.

C. Create a prefix for the joined class.

D. Add an association rule to match key values.

ANSWER: C D

Explanation:

A class join in a Pega report definition brings properties from a related class into the report query. You must create a prefix for the joined class because the prefix qualifies references to properties from that class. This provides an unambiguous way to select, filter, sort, and display properties when the primary and joined classes might contain properties with the same name. For example, a property reference can use the join prefix to identify that its value comes from the related class rather than from the report's primary class.

You must also add an association rule to match key values. The association defines the relationship between the primary class and the joined class, including the key-property mappings used to locate related instances. The report definition relies on that association to generate the join relationship and retrieve the appropriate data. Together, the association supplies the relationship logic, while the prefix makes the joined-class properties usable in the report configuration. See Pega's documentation on creating report definitions and the Pega Academy topic on reporting.

QUESTION NO: 2

Which configuration allows you to control which user roles can add attachments to a case?

A. Configure a privilege on the attachment category.

B. Enable attachment-level security on the attachment category.

C. Update the access control setting for adding attachments in the Access Manager.

D. Configure a when rule on the attachment category.

ANSWER: A

Explanation:

Configure a privilege on the attachment category. This is the appropriate configuration because an attachment category can specify a privilege that is required to add an attachment in that category. Privileges are authorization capabilities that can be granted to an access role through the role's privilege configuration. Consequently, the application can allow users with one access role to add attachments of a particular category while preventing users whose roles do not include that privilege from doing so.

This approach separates attachment authorization from the case UI itself and makes the rule reusable. For example, a category intended for sensitive supporting documents can require a dedicated privilege; that privilege is then assigned only to the roles that are authorized to submit those documents. At run time, Pega evaluates the user's access roles and their granted privileges when the user attempts to add an attachment under the category. This provides role-based control at the attachment-category level and is consistent with Pega's general security model, in which access roles receive privileges and privileges are checked by application rules. See Pega documentation on controlling access to case attachments and access roles.

QUESTION NO: 3

Several development teams work on different enhancements. The release date for each enhancement is uncertain. Which two options allow each team to keep its work separate? (Choose Two)

A. Set up a branch ruleset for each team.

B. Create a new application for each team.

C. Create a new ruleset version for each team.

D. Create a production ruleset for each team.

ANSWER: A C

Explanation:

Setting up a branch ruleset for each team is appropriate because a branch provides an isolated development space for rules that are being changed for a specific feature or enhancement. Teams can develop, test, and review their changes independently, then merge approved branch contents into the destination ruleset when that enhancement is ready to proceed. This is particularly useful when release schedules are uncertain because unfinished work remains outside the shared ruleset until it is intentionally merged.

Creating a new ruleset version for each team also separates rule changes. A ruleset version is the unit in which Pega stores and manages rules, and developers can save changes to an unlocked version without altering an earlier, locked version used by a current release. Using separate versions for independently managed enhancement work provides controlled rule management and lets teams retain their changes until the appropriate release process incorporates them. Pega's guidance on branches and ruleset versioning is available in the Pega documentation for creating branches and the Pega documentation for ruleset versioning.

QUESTION NO: 4

In which two of the following situations would you simulate an integration? (Choose Two)

A. The connector is configured to use global resource settings.

B. The service is not available yet.

C. The service has slow response times.

D. You need to test each flow path in the case processing.

ANSWER: B D

Explanation:

Simulating an integration is appropriate when **the service is not available yet** because it allows the application team to continue building and testing the case without waiting for an external system, endpoint, or service contract to be deployed. A simulated connector can return representative response data locally, so the application can exercise the expected integration behavior during development.

Simulation is also appropriate when **you need to test each flow path in the case processing**. By configuring simulated responses that represent different business outcomes, such as a successful result, a validation outcome, or an exception condition, developers can verify that the case follows the intended routing and processing paths. This makes integration-dependent case behavior repeatable and testable without relying on an external system to consistently produce every required response scenario. Pega integration testing supports using simulated connector responses to isolate the application from external dependencies and validate its processing logic. See Pega Academy: Creating connector rules and Pega Academy: Testing an application.

QUESTION NO: 5

You are creating a report that lists all open Personal Injury cases for an Auto Claim case. What three options do you use to configure the join for this report? (Choose Three)

A. Create the report in the class group.

B. Specify a join for the Personal Injury class.

C. Specify the join type to include all rows in the Personal Injury class.

D. Specify the join type to only include matching rows.

E. Specify a join for the Auto Claim class.

F. Create the report in the Auto Claims class.

ANSWER: B D F

Explanation:

Create the report in the Auto Claims class. is appropriate because the Auto Claim case is the primary context of the report. The report definition's applies-to class establishes the primary records and properties that are available before the join is evaluated. **Specify a join for the Personal Injury class.** adds the related case class as the secondary source, allowing the report to retrieve Personal Injury case data that is associated with each Auto Claim record. The join conditions identify the relationship between the Auto Claim and Personal Injury instances, commonly by matching the appropriate case or parent/cover key property. **Specify the join type to only include matching rows.** creates an inner-join-style result: a row is returned only when the Auto Claim record and the related Personal Injury record satisfy the configured join condition. This produces the related Personal Injury cases for the Auto Claim rather than retaining unrelated records. The report can then apply its report filter to limit the joined Personal Injury cases to those that are open. Pega report definitions support joins to combine data from a primary class with data from a related class, and the join type controls which matched records are returned. See Pega documentation on creating report-definition joins and Pega Academy reporting guidance.

QUESTION NO: 6

You create an activity that updates sensitive customer information. Only users with a CustomerUpdate privilege can run the activity. Which two configurations satisfy this requirement? (Choose Two)

A. Add the CustomerUpdate privilege to the activity.

B. Add the CustomerUpdate privilege to the appropriate access role.

C. Add the CustomerUpdate privilege to the Operator ID record.

D. Configure CustomerUpdate as a node-level security setting.

ANSWER: A B

Explanation:

Add the CustomerUpdate privilege to the activity and **add the CustomerUpdate privilege to the appropriate access role**. A privilege on an activity provides the authorization requirement that Pega evaluates before the activity can be executed.

An access role grants the required privilege to authorized users through their access group. Users whose current access roles do not include the privilege cannot execute the protected activity. This approach provides reusable, role-based authorization and supports least-privilege security. See Pega documentation on authorizing access with privileges.

QUESTION NO: 7

You want to restrict access to a workbasket to certain users. How do you configure this requirement?

A. Add a skill threshold to the workbasket.

B. Add the access role for allowed users to thp workbasket record.

C. Add a privilege to the workbasket record and to the appropriate access role.

D. Add the access group for allowed users to the workbasket record.

ANSWER: C

Explanation:

Add a privilege to the workbasket record and to the appropriate access role. In Pega, a workbasket can be secured by associating a privilege with its Data-Admin-WorkBasket instance. A user must have that privilege through an access role in their current access group to open, view, or process assignments in the protected workbasket. This provides authorization-based control: administrators grant the privilege only to the roles used by the intended population, and users receive access according to their role configuration rather than through an individually maintained list on the workbasket. Privileges are designed for this type of targeted authorization check and can be reused consistently across rules and security configurations. Configure the privilege on the Security tab of the workbasket record, then add the same privilege to the relevant access role by using the role's privileges configuration. At run time, Pega evaluates the operator's access roles and permits workbasket access when the required privilege is present. This approach supports least-privilege access and makes security administration manageable as users and teams change. See Pega documentation on authorizing access with privileges and the Pega access-control model.

QUESTION NO: 8

The ruleset list for an application consists of the following rulesets, ordered from highest to lowest:

A rule with an Apply to: class of TGB-HR-SelfService-Work-TimeOff references a rule named EnterEmphyeeDetails. The four instances of EnterEmployeeDetails in the rules cache are shown in the following table.

Which instance of EnterEmployeeDetails is chosen during rule resolution?

A. TGB-HR-SelfService-Work-TimeOff .EnterEmployeeDetails (SelfService:01-01-02)

B. TGB .EnterEmployeeDetails (TGB:01-01-02)

C. TGB-HR-SelfService-Work .EnterEmployeeDetails (SelfService:01-01-01) .

D. TGB-HR-SelfService-Work-TimeOff.EnterEmployeeDetails (SelfService:01-01-01)

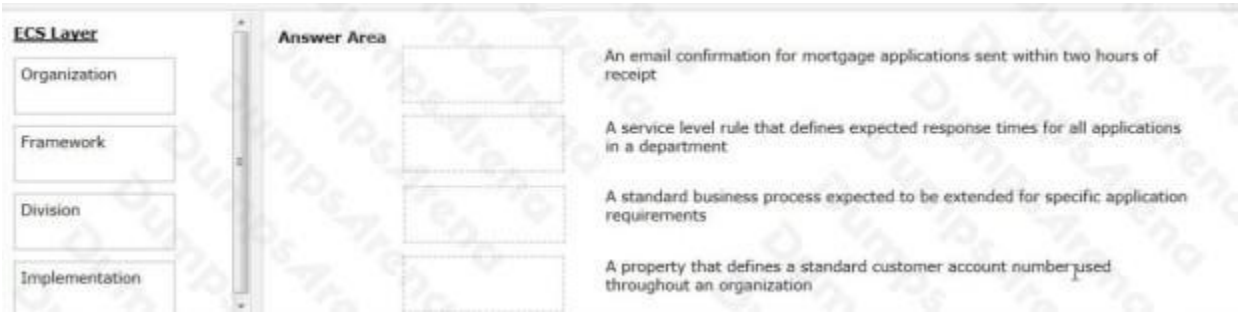
ANSWER: A

Explanation:

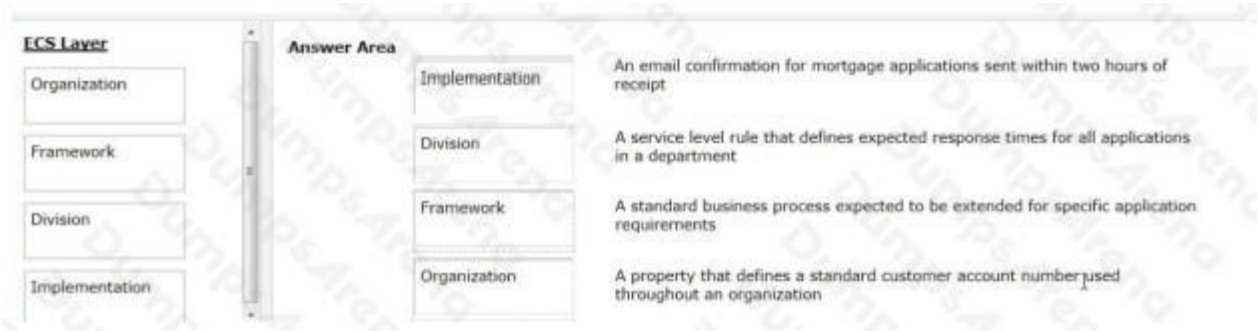
`TGB-HR-SelfService-Work-TimeOff .EnterEmployeeDetails (SelfService:01-01-02)` is selected because rule resolution first seeks an available instance in the class of the requesting rule, which is `TGB-HR-SelfService-Work-TimeOff`. Pega evaluates candidate rules using the application's ruleset stack and selects the highest available ruleset version that is valid for the request. Two cached instances use that exact Apply to class and the `SelfService` ruleset: versions `01-01-02` and `01-01-01`. Since `01-01-02` is the higher version, it takes precedence over the lower-version instance. This resolution behavior enables an application to use the most current rules available in its configured ruleset stack while retaining prior versions for ruleset versioning and maintenance. Pega's rule resolution process considers class inheritance only when an applicable rule is not found at the more specific class level, so the exact-class candidate remains the appropriate resolution result. See Pega Academy's Rule resolution topic and its overview of rulesets for the relationship between ruleset stacks, versions, and rule selection.

QUESTION NO: 9 - (DRAG DROP)

Organize each rule in the appropriate layer of the Enterprise Class Structure (ECS).



ANSWER:



Explanation:

The Enterprise Class Structure organizes reusable Pega rules according to their intended scope and the specialization level at which they apply. A property defining a standard customer account number belongs in the Organization layer because it represents a common enterprise data concept intended for consistent use throughout the business. Establishing such shared properties at the organization scope supports a uniform data model and lets all divisions, frameworks, and implementations inherit the same definition.

A service-level rule governing expected response times for every application in a department belongs in the Division layer. A division is the right scope for rules that are shared by multiple applications serving one business unit or department, while still allowing policies to differ across other parts of the enterprise.

A standard business process that is deliberately designed to be extended for application-specific requirements belongs in the Framework layer. Framework rules provide a reusable industry, product, or business-pattern foundation. Implementing applications inherit that foundation and specialize it where their own needs require additional behavior or configuration.

The mortgage-application email confirmation belongs in the Implementation layer because it is concrete application behavior for a particular implemented solution. The two-hour confirmation requirement is part of the operational experience delivered by that application. This arrangement preserves inheritance from broadly reusable definitions while locating specialized behavior where it is maintained and released. Pega's documentation describes the class hierarchy and inheritance concepts used to organize reusable rules in an application: Pega Documentation. Additional platform learning resources are available through Pega Academy.

QUESTION NO: 10

In a flow, an integrator shape calls a connector that writes data to a system of record. The system of record is not currently available.

How do you simulate the service call invoked from the integrator shape?

A. Leave the flow in draft mode and simulate integration values in a data transform.

B. Create a simulation activity for the connector.

C. Configure global resource settings for the connector.

D. Simulate the data page using a data transform.

ANSWER: B

Explanation:

Create a simulation activity for the connector. A connector simulation activity lets the application execute the flow path without making a live request to the unavailable system of record. The activity supplies the expected simulated response on the clipboard, allowing the Integrator shape and the downstream flow logic to be tested under controlled conditions. This is particularly useful for a connector that performs a write operation, because the test can validate request preparation, response handling, error paths, and case processing without changing data in the external system. The simulation is associated with the connector invocation, so it models the result at the appropriate integration boundary rather than requiring changes to the business flow itself. A simulation activity can also return different representative results as needed for test scenarios, such as a successful update or an application-level status returned by the external service. Pega's connector architecture is designed to separate application logic from communication with external systems, which makes connector-level simulation suitable when the target service is unavailable. See Pega documentation on connector rules and Pega Academy's overview of connecting external systems.

QUESTION NO: 11

A data page retrieves a list of currency exchange rates from an external REST service. The rates are the same for all users and must be refreshed once each business day.

Which data page scope do you configure?

A. Thread

B. Requestor

C. Node

D. Case

ANSWER: C

Explanation:

Node is correct because the exchange-rate list is shared by all requestors on the same node and does not depend on an individual user or case. A node-scoped data page is loaded once for the node and can be reused by all requestors that require the same data, reducing repeated connector calls.

The refresh strategy can be configured so that the node-scoped page is reloaded after the required daily interval. Thread scope would create separate data for each requestor thread, while requestor scope would unnecessarily create a separate cached instance for each user. See the Pega documentation on data pages and Pega Academy data pages.