DUMPSARENA

Certified Pega Senior System Architect 23 Exam

Pegasystems PEGACPSSA23V1

Version Demo

Total Demo Questions: 17

Total Premium Questions: 172

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	31
Topic 2, Case Management	25
Topic 3, Data and Integration	30
Topic 4, User Experience	25
Topic 5, Security	24
Topic 6, Reporting	8
Topic 7, Background Processing	4
Topic 8, DevOps	25
Total	172

QUESTION NO: 1

Your application contains the following versions of a service level named AuthorizeClaim.

Class	Ruleset	Version	Circumstance
MyCo-AccountManage-Work	AccountManage	01-01-01	None
MyCo-AccountManage-Work	AccountManage	01-01-05	.Status="Silver"
MyCo-AccountManage-Work	AccountManage	01-01-05	.Status="Gold"
MyCo-AccountManage-Work	AccountManage	01-01-05	.Status="Platinum"
MyCo-AccountManage-Work-AccountOpen	AccountManage	01-01-05	.Status="Silver"
MyCo-AccountManage-Work-AccountOpen	AccountManage	01-01-05	.Status="Gold"
MyCo-AccountManage-Work-AccountOpen	AccountManage	01-01-05	.Status="Platinum"

The application is being updated with a new version of the AccountManage ruleset. As part of the update, the service level is no longer circumstanced for AccountOpen cases. The update must retain the circumstancing for the other case types in the application.

Which option do you use to reset the circumstancing for the AccountOpen case type?

A. Copy the non-circumstanced rule to the AccountOpen class and select the base rule option.

B. Make the circumstanced rules in the AccountOpen class unavailable.

C. Withdraw the circumstanced rules in the AccountOpen class.

D. Block the circumstanced rules in the AccountOpen class.

ANSWER: C

Explanation:

Withdraw the circumstanced rules in the AccountOpen class. Withdrawing is the appropriate rule-resolution mechanism when a rule instance that was available in an earlier ruleset version must no longer be selected in a newer version. In this situation, the withdrawal applies specifically to the AuthorizeClaim rule instances circumstanced for the AccountOpen class. When rule resolution encounters the withdrawn AccountOpen circumstanced instance, it does not use that specialized version and can resolve to the applicable non-circumstanced base service-level rule instead. This achieves the required reset for AccountOpen cases without removing or altering circumstanced AuthorizeClaim instances for other case types. The withdrawal is created in the new AccountManage ruleset version, preserving the historical rule record in prior versions while changing behavior for applications that include the updated ruleset. This is particularly useful during application evolution because it provides an explicit, versioned indication that the specialized rule should cease being used, while allowing the remaining valid specialization hierarchy to continue operating. Pega documents withdrawal as a rule-management capability that prevents an earlier rule instance from being found by rule resolution in later ruleset versions. See Pega documentation on ruleset versioning and Pega documentation on rule resolution.

QUESTION NO: 2

MyCo insurance company completes the development phase of its application and decides to start the testing phase of the application in a testing environment. To migrate the application, the development team created an instance of the Rule-Admin-Product class that includes the application instances to migrate.

Which two components does the Rule-Admin-Product instance include by default when you generate an archive file? (Choose two.)

A. Checked-in rules

B. MyCo- class instances

C. Data- class instances

D. Checked-out rules

ANSWER: A B

Explanation:

Checked-in rules and **MyCo- class instances** are included by default in an archive generated from a Rule-Admin-Product rule. A product rule is Pega's primary mechanism for defining the contents of a deployment package. Its ruleset contents represent versioned, checked-in rules, ensuring that the target environment receives stable rule versions that are available to the application at runtime. This supports controlled promotion of an application from development to a test environment.

The archive also includes instances in the implementation-class hierarchy identified by the organization's class prefix, such as `MyCo-`. These instances can include application-specific configuration, data, and other records that belong to the implementation and are required for the migrated application to operate consistently in the target environment. Including the implementation classes by default helps move the application's owned artifacts without requiring every relevant class instance to be specified individually.

Teams can further tailor a product rule by selecting ruleset versions and adding, excluding, or otherwise controlling class instances as needed for a particular deployment. See Pega documentation on creating a product rule and exporting an application.

QUESTION NO: 3

A company with multiple applications spanning different business units wants to send a standard confirmation email to customers whenever a case is created.

Which is the appropriate layer of the Enterprise Class Structure (ECS) for the email correspondence?

A. Unit

B. Organization

C. Implementation

D. Division

ANSWER: B

Explanation:

Organization is the appropriate ECS layer because the confirmation email is a common enterprise capability: it is intended for use by multiple applications and across different business units. In Pega, rules placed at a higher ECS layer are available for reuse by lower layers through class inheritance. Defining the correspondence rule in the organization layer establishes one centrally governed version of the standard customer confirmation, avoiding duplicate rule definitions and helping maintain consistent wording, branding, and behavior throughout the enterprise.

The organization layer represents assets that apply broadly across the whole organization rather than to one particular line of business, operational unit, or application implementation. A correspondence rule can be created at this layer and then referenced by the case processing in the applications that need to send it. If an application later requires a specialized message, it can specialize the inherited rule at an appropriate lower layer while retaining the enterprise default. This approach follows ECS's goal of maximizing reusable, shared rules while allowing controlled specialization where a genuine business need exists.

For an overview of ECS and inheritance-based reuse, see Pega Academy: Enterprise Class Structure. For correspondence configuration concepts, see Pega Academy: Correspondence.

QUESTION NO: 4

A travel reservation servicing case includes a service level for responding to requests. The service level intervals vary according to passenger status, class of service, and fare type.

Which implementation satisfies this requirement?

A. Create a multivariate circumstanced rule and a when rule.

B. Create a multivariate circumstanced rule, a circumstance definition, and a circumstance template.

C. Create a single property circumstanced rule and a when rule.

D. Create a single property circumstanced rule.

ANSWER: B

Explanation:

Create a multivariate circumstanced rule, a circumstance definition, and a circumstance template. The required service-level interval depends on the combined values of three independent business factors: passenger status, class of service, and fare type. Multivariate circumstancing is designed for this situation because it selects the appropriate rule instance according to a combination of property values, rather than only one property value or a separately evaluated condition. In Pega, the circumstance template defines the properties that participate in the multivariate circumstance and their permitted values. The circumstance definition then identifies the specific combinations for which specialized rule instances are needed. The specialized service-level rules can consequently provide the correct response interval for each applicable passenger and booking scenario, while the base rule remains available as a fallback when no defined combination applies. This approach keeps the service-level policy declarative, maintainable, and directly aligned to the business dimensions that determine the deadline. Pega documentation describes multivariate circumstancing and the use of templates and definitions in Multivariate circumstancing and provides the broader rule-resolution context in Rule resolution.

QUESTION NO: 5

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

A. Create a new application for each team.

B. Create a new ruleset version for each team.

C. Set up a branch ruleset for each team.

D. Create a production ruleset for each team.

ANSWER: B C

Explanation:

Creating a new ruleset version for each team and setting up a branch ruleset for each team provide both version-level separation and isolated parallel development. A ruleset version is the unit in which Pega stores and versions application rules. Assigning each enhancement stream its own version prevents one team's in-progress rule changes from being mixed with another team's work in the same ruleset version. This is particularly useful when the teams' release schedules are independent.

Branch rulesets add a dedicated development workspace on top of the base ruleset versions. Each team can create, update, and test rules in its own branch without immediately changing the rules used by other teams or the shared application. When an enhancement is ready, its branch can be merged into the appropriate ruleset version; an enhancement with a delayed release can remain isolated until it is ready. Together, separate ruleset versions and per-team branch rulesets support controlled parallel development, independent release timing, and a clear merge path. Pega describes branch rulesets and their merge workflow in the Pega Platform documentation on branch rulesets and the Pega Academy application development topic.

QUESTION NO: 6

An application contains the class group MyCo-HR-SelfService-Work. There are two classes derived from Work- class:

If a report is created in MyCo-HR-SelfService-Work, what instances will the report return?

A. Instances of MyCo-HR-SelfService-Work-TimeOff and MyCo-HR-SelfService-Work-Expense, unless they are stored in different database tables

B. Instances of all Work- derived classes

C. Instances of MyCo-HR-SelfService-Work

D. Instances of MyCo-HR-SelfService-Work-TimeOff and MyCo-HR-SelfService-Work-Expense

ANSWER: A

Explanation:

Instances of MyCo-HR-SelfService-Work-TimeOff and MyCo-HR-SelfService-Work-Expense, unless they are stored in different database tables, is correct. A class group is used to associate classes that share a database table. When a report definition is created in the class group, Pega queries the table associated with that class group. Consequently, the report can return persisted work instances whose concrete classes are members of that group, including the TimeOff and Expense work classes in this scenario. This makes a class-group report useful for reporting across related case types without creating separate reports for each case class.

The database mapping remains essential to the report's scope. Pega reporting operates on the data source table available to the report class. Therefore, instances are included when the relevant derived work classes are persisted in the table represented by the MyCo-HR-SelfService-Work class group. If a derived class has been mapped to a separate table, its instances are not part of that class-group table query and require reporting against the appropriate data source. Pega's data-modeling guidance describes how class groups organize classes that share a table, while reporting guidance explains that report definitions retrieve data from the report class's mapped storage. See Pega documentation on creating class groups and Pega Academy reporting topic.

QUESTION NO: 7

A data page is used to retrieve data from an external system. If an error occurs, you want to display a message to the user and send an email to the system administrator.

How do you implement this requirement?

A. Configure an activity's input source as the error message and the output as an email.

B. Configure an error handling process that displays the error message and sends an email.

C. Reuse the out-of-the-box ConnectionProblem error handling flow on the Service tab for the connector.

D. Create an error handler data transform that adds an error message to the data page and sends an email.

ANSWER: B

Explanation:

Configure an error handling process that displays the error message and sends an email. A data page that obtains information through a connector can encounter connection, timeout, authentication, or response-processing failures while loading its data. Configuring a dedicated error-handling process provides a single, controlled place to respond to that failure. The process can add a meaningful, user-safe error message to the case or UI context so that the user understands that the requested information is currently unavailable. It can also use Pega's email capability to notify the system administrator, including diagnostic context such as the affected integration, error details, and case identifier where appropriate. This design separates normal data retrieval from exception processing, supports consistent operational monitoring, and allows the error-handling behavior to be maintained independently of the connector's successful-response mapping. It also ensures both required actions are coordinated as part of the same failure response rather than relying on a client-side message alone. See Pega documentation search results for data page error handling and Pega Academy's overview of data pages.

QUESTION NO: 8

Which two factors are used by Pega rule resolution to locate an applicable rule instance? (Choose Two)

A. The rulesets and versions in the application ruleset stack

B. The class hierarchy of the class where the rule is requested

C. The label specified on the rule form

D. The operator ID of the developer who last updated the rule

ANSWER: A B

Explanation:

Pega uses the application ruleset stack to determine which rulesets and versions are available to the current application. Rules in the appropriate ruleset context can override rules from lower-precedence rulesets according to the application configuration.

Pega also considers class inheritance when searching for an applicable rule. If a rule is not found in the class where it is requested, Pega can search eligible parent classes according to the class hierarchy. A rule's label or the name of the operator who created it does not determine rule resolution.

QUESTION NO: 9

Which two methods allow you to identify the Pega Platform application type? (Choose Two)

A. View the Settings tab of the case designer.

B. View the Application layers widget for the ruleset stack.

C. View the Application definition rule.

D. View the portals used by the application.

ANSWER: B C

Explanation:

Viewing the Application layers widget for the ruleset stack and viewing the Application definition rule are the appropriate ways to identify an application’s type and architectural context. The Application layers widget provides a visual representation of the application stack, showing the implementation application, any framework layers on which it is built, and the underlying Pega Platform layer. This lets an architect quickly determine whether the application is a stand-alone application or is built on a Pega industry or enterprise framework. The Application definition rule provides the authoritative configuration details for an application, including its ruleset stack, version, built-on application, and other structural settings. Reviewing this rule therefore reveals the application’s relationship to its parent framework or platform application and supports accurate identification of its type. These views are particularly useful during application maintenance, upgrade planning, and troubleshooting because they expose the layering model that determines rule resolution and reuse. For more information, see Pega documentation on application rules and Pega Academy: Application layering.

QUESTION NO: 10

An assignment service-level agreement (SLA) is configured with the following details:

? Initial urgency: 20

? Assignment ready: Timed delay of 1 hour

? Goal: 5 hours and increase urgency by 10

? Deadline: 8 hours and increase urgency by 20

? Passed deadline: 2 hours, increase urgency by 20, and limit events to 5

Assuming no other urgency adjustments, what is the assignment urgency 16 hours after the case reaches the assignment.

A. 100

B. 130

C. 70

D. 90

ANSWER: B

Explanation:

130 is correct. The assignment starts with an initial urgency of 20. When the goal is reached at five hours, the SLA adds 10, bringing urgency to 30. At the deadline of eight hours, it adds another 20, resulting in urgency of 50.

After the deadline, the passed-deadline interval is two hours. By the 16-hour point, four passed-deadline events have occurred: at 10, 12, 14, and 16 hours. Each event increases urgency by 20, for an additional 80 urgency points. The configured event limit is five, so it does not prevent any of these four events from running. The final urgency is therefore 20 + 10 + 20 + 80 = 130.

The Assignment ready value controls when the assignment becomes available for processing; it does not add an extra urgency adjustment. Pega uses SLA goal, deadline, and recurring passed-deadline events to automatically adjust assignment urgency and support appropriate work prioritization. See Pega documentation on service-level agreements and Pega Academy's Service-level agreements topic.

QUESTION NO: 11

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

A. Select the type of match for key values.

B. Add an association rule to match key values.

C. Create a prefix for the joined class.

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

ANSWER: A C

Explanation:

When configuring a class join in a report definition, you must select the type of match for key values and create a prefix for the joined class. The match type controls how the report combines instances from the report's primary class with instances in the joined class based on the configured join conditions. For example, the configuration determines whether the result includes only records with matching values or also retains records from the primary class when no related instance is found. This choice is essential because it defines the relationship behavior and resulting report rows.

A prefix is also required for the joined class so that its properties can be referenced unambiguously throughout the report definition. The prefix qualifies properties selected from the joined class, including when those properties are used in columns, filters, sorting, or additional calculations. This is particularly important when the primary and joined classes contain properties with the same name. Together, the match selection and joined-class prefix establish both the join behavior and a reliable namespace for using data from the related class. For Pega reporting guidance, see Pega Academy: Report Definition rules and Pega documentation on class joins in report definitions.

QUESTION NO: 12 - (SIMULATION)

You perform a major skim on the ruleset ABC:02-02-01 to ABC:03-01-01.

In the Answer area, select the ruleset versions that the skim operation uses to copy rules into ABC:03-01-01.

ANSWER: See the explanation for the answer

Explanation:

Select No for every listed ruleset version:

`ABC:02-01-01` — No
`ABC:01-02-01` — No
`ABC:02-01-05` — No

A major skim creates `ABC:03-01-01` using rules from the highest minor version in the current major version, which is `ABC:02-02-01`. The listed `02-01-xx` versions are lower minor versions, and `01-02-01` belongs to a prior major version, so none are used as the source for this skim.

QUESTION NO: 13

Which two statements about keyed data pages are true? (Choose Two)

A. A keyed data page is required for all data pages.

B. A keyed data page can use a report definition.

C. A keyed data page can return multiple embedded pages.

D. A keyed data page can have multiple keys.

ANSWER: C D

Explanation:

A keyed data page can return multiple embedded pages because it uses a page-group structure. A page group holds an embedded page for each distinct key, allowing an application to retrieve a particular page directly by supplying its associated key. For example, a data page can retain separate embedded customer pages under different customer identifiers. This structure is useful when the application needs efficient keyed access to several related instances rather than a single page or an ordered list.

A keyed data page can have multiple keys because the page group can contain many key-and-page entries at the same time. Each key identifies one embedded page within the data page, so the data page can cache and provide access to multiple results independently. The key is part of the page-group addressing syntax and enables Pega to locate the required embedded page. Pega data pages support Page, List, and Page Group structures; the Page Group structure is the one used for keyed access and for maintaining multiple embedded pages indexed by keys. See Pega Academy: Data pages and Pega Platform documentation: Data pages.

QUESTION NO: 14

An application includes the following set of circumstanced instances of a decision table.

Class	Ruleset	Version	Circumstance
MyCo-AccountManage-Work-AccountOpen	AccountManage	01-01-01	None
MyCo-AccountManage-Work-AccountOpen	AccountManage	01-01-05	.Status="Silver"
MyCo-AccountManage-Work-AccountOpen	AccountManage	01-01-05	.Status="Gold"
MyCo-AccountManage-Work-AccountOpen	AccountManage	01-01-15	.Status="Platinum"
MyCo-AccountManage-Work-AccountOpen	AccountManage	01-01-15	.Status="Silver"

You update the application and increment the AccountManage ruleset version to 01-02-01. As part of this update, the circumstance .Status="Silver" is no longer needed by the application.

How do you implement this change?

A. Create a new version of the circumstanced decision table in AccountManage:01-02-01 and select the Base rule option.

B. Create a new version of the circumstanced decision table in AccountManage:01-02-01 and set the availability of the rule to Withdrawn.

C. Create a new version of the circumstanced decision table in AccountManage:01-02-01 and set the availability of the rule to Blocked.

D. Do nothing. Rule resolution does not consider rules in a lower minor version of a ruleset.

ANSWER: B

Explanation:

Create a new version of the circumstanced decision table in AccountManage:01-02-01 and set the availability of the rule to Withdrawn. A withdrawn rule explicitly removes the inherited circumstanced instance from use at the newer ruleset version. In this case, withdrawing the instance for `.Status="Silver"` ensures that rule resolution does not continue to select the matching instance saved in an earlier available version of the AccountManage ruleset. The higher-version withdrawn instance acts as the intentional retirement point for that circumstance while preserving the rule history in prior versions.

This is particularly important because ruleset versioning supports application evolution: a newer ruleset version can still access eligible rules in earlier versions unless a rule's availability is explicitly controlled. By creating the matching circumstanced instance in the updated ruleset and marking it Withdrawn, the application cleanly expresses that the Silver-specific behavior must no longer be available from this version onward. Pega rule resolution evaluates circumstanced rules using the requestor context and the rule cache, so availability settings are part of determining whether a resolved rule instance can be used. See Pega Academy's Rule resolution and Ruleset versioning guidance.

QUESTION NO: 15

An assignment service-level agreement (SLA) is configured with the following details:

â™¦ Initial urgency: 20

â™¦ Assignment ready: Timed delay of 2 hours

â™¦ Goal: 5 hours and increase urgency by 10

â™¦ Deadline: 2 hours and increase urgency by 25

â™¦ Passed deadline: 1 hour, increase urgency by 5, and limit events to 6

The case reaches the assignment at 9 AM on Wednesday.

Assuming no other urgency adjustments, what is the assignment urgency 6.5 hours after the case reaches the assignment?

A. 85

B. 95

C. 75

D. 80

E. 20

ANSWER: E

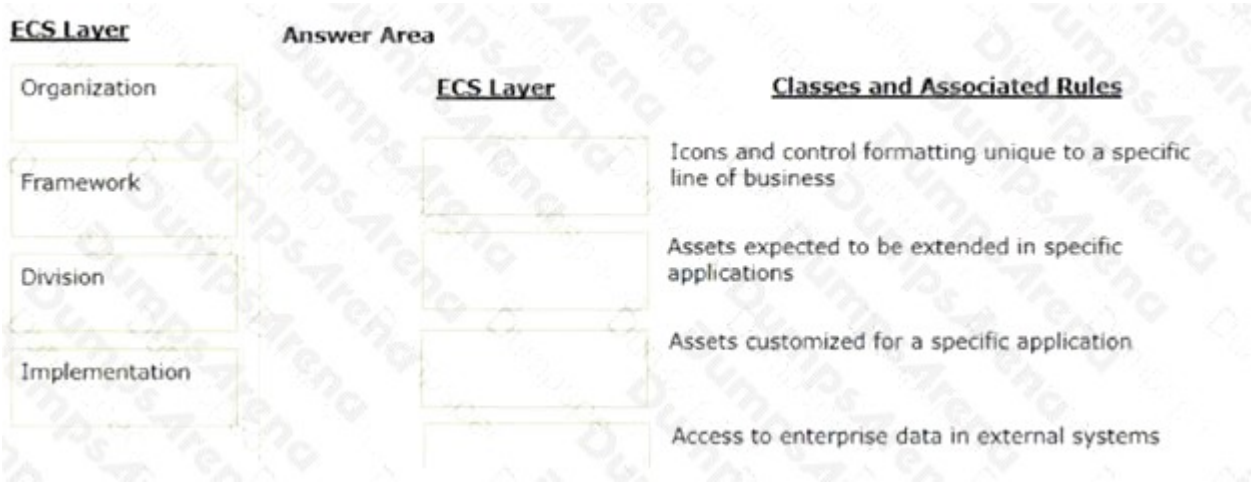
Explanation:

20 is correct. The Assignment ready setting postpones availability of the assignment for two hours. Therefore, an assignment reached at 9 AM becomes ready at 11 AM. SLA processing for the remaining milestones proceeds from that ready time. The

five-hour Goal interval consequently expires at 4 PM. At 3:30 PM, which is 6.5 hours after the assignment was reached, the Goal has not yet occurred. Neither the Deadline nor Passed deadline processing has started, because those events occur later in the SLA sequence. The assignment consequently retains its configured initial urgency of 20 at that point. The limit of six Passed deadline events does not affect the calculation because no Passed deadline event has occurred. This timing model is important when calculating urgency: Assignment ready defines when work becomes available, and subsequent SLA milestones are evaluated based on the assignment becoming ready. Pega's SLA configuration supports an Assignment ready interval followed by Goal, Deadline, and repeating Passed deadline escalation events. See the Pega Academy material on service-level agreements and the Pega documentation on service-level agreements.

QUESTION NO: 16 - (SIMULATION)

Organize the classes and associated rules in the appropriate Enterprise Class Structure (ECS) layer.



ANSWER: See the explanation for the answer

Explanation:

Assign the classes and rules to the ECS layers as follows:

Organization — Access to enterprise data in external systems
Framework — Assets expected to be extended in specific applications
Division — Icons and control formatting unique to a specific line of business
Implementation — Assets customized for a specific application

Organization-level rules support enterprise-wide reuse, division rules apply to a line of business, framework assets are intended for extension, and implementation-layer rules contain application-specific customization.

QUESTION NO: 17

A city resident can report potholes to the Department of Transportation by logging in to a mobile Pega Platform application that utilizes the Pega API.

Which two Pega API interactions do you use to facilitate this? (Choose two.)

A. Submit the report using POST/cases

B. Access the related case type to report using GET/casetypes

C. Update the report using GET/cases

D. Log into the application using PUT/authenticate

ANSWER: A B

Explanation:

Submit the report using POST/cases is correct because a pothole report is modeled as a Pega case. The POST `/cases` endpoint creates a new case from the mobile client, using the selected case type and the report data supplied by the resident, such as the location, description, and any supporting details. Pega then returns information about the newly created case, allowing the application to present confirmation or continue the case workflow.

Access the related case type to report using GET/casetypes is also correct because the mobile application can retrieve the case types that are available to the authenticated user. This enables the client to identify and select the appropriate pothole-report case type before submitting the creation request. Using the available case-type metadata helps the application initiate the correct business process and present suitable case choices to the resident. Together, these interactions support discovering the reportable case type and creating the resulting service-request case through the Pega API. See the Pega API documentation and the Pega API reference.