1. Availability types?Difference between withdrawn and Blocked?

A rule is **available** if the value of the pyRuleAvailable property is set to Yes or Final (and is not blocked; see below). Developers determine this value as they create or save a rule.

A green icon at the upper right of the rule form indicates the current setting. When presented in reports, the icon is brown.

This property may have any of five values:

◎ Yes ⊙	A circle and dot indicates that this rule is available. Rules with a Yes value are visible to rule resolution processing and can be executed.
● No/Draft Mode ○	An empty circle indicates a rule that is **not** available. Set the Availability of a rule to No/Draft Mode to cause the rule to become invisible to the rule resolution algorithm for all users (including yourself), and to bypass validation of non-key fields. The No/Draft Mode setting is useful in experimentation and troubleshooting to assess the effect of two rule versions.
⊗ Blocked ⊗	A circle within an X indicates that this rule has Availability set to Blocked. Set the value of this property to Blocked if you want rule resolution processing to halt (with no rule found) when it encounters this rule. The rule form colors change from greens to grays. This is a stronger form of No/Draft, because it affects all lower-numbered versions of the rule, and versions (in other RuleSets) that are below this version in the user's RuleSet list. A blocked rule does not prevent rule resolution from finding (and running) higher-numbered versions in the same RuleSet, or finding rules with the same visible key in a different RuleSet that is higher on the RuleSet list.
⇥ Final ⇥	An arrow and barrier in the key area indicates that this rule has Availability set to Final. Set this value of this property to Final if you want this rule to be available (equivalent to Yes), but want to prevent others from overriding this rule in any other RuleSet. A final rule can be superceded by a higher-numbered version in the same RuleSet, but not by any version in any different RuleSet.
Withdrawn	This image indicates that the rule has Availability set to Withdrawn. A withdrawn rule is never selected by rule resolution. In addition, a withdrawn rule masks from rule resolution any other rules which meet all of these tests: ▪ The other rule belongs to the same RuleSet ▪ The other rule belongs to the same major version of the RuleSet, but a lower minor version or lower patch version ▪ The other rule has the same Applies To class (if relevant and other key parts match) ▪ If circumstance-qualified, is qualified with the same circumstance type and value.

2. What is skimming?Will withdrawn rule be a part of Skimming?

Skimming simplifies the topmost rule versions in a RuleSet after multiple iterative development cycles. For example, if the highest version was previously 02-21-06:

- After a **major** skim, the skimmed version number is 03-01-01.
- After a **minor** skim, the skimmed version number is 02-22-01.

Rules in lower versions such as 02-21-05, 02-18-53, or 01-88-15 of the same rule are not copied by the skim operation.

The skim automatically creates the new RuleSet version instance corresponding to the major, minor or patch version above the current version. It then copies rules from the current version (selecting only those in the highest-numbered version) into the new major or minor version.

For a major skim, rules with availability of Blocked in the source RuleSet versions are not copied. (For a minor skim, blocked rules are copied into the resulting RuleSet version.

Skimming copies **rules with availability Yes, No/Draft, Withdrawn, and Final from the source RuleSet version to the skimmed RuleSet** version. The results of a skim operation exclude any rules with availability Blocked, and also any rules that are in lower versions and blocked by the blocked rule.

Class rules, library rules, Access of Rule to Object rules, RuleSet Name rules and the RuleSet Version rules themselves have no associated RuleSet version. They are unaffected by the skim operation.

3. What is Inception?Several phases of Project?

4. What are the different types of Declarative rules you have worked on?

Declarative processing simplifies your application and reduces the number of activities you need to create. The declarative processing facilities support operation of six rule types:

- Constraints rules (Rule-Declare-Constraints rule type)
- Declare Expression rules (Rule-Declare-Expressions rule type)
- Declare Index rules (Rule-Declare-Index rule type)
- Declare OnChange rules (Rule-Declare-OnChange rule type)
- Declare Trigger rules (Rule-Declare-Trigger rule types)
- Declare Pages rules (Rule-Declare-Pages rules type)

On the Application Explorer and Rules Explorer display, four of these rule types belong to the Decision category. Declare Index rules support reporting and belong to the SysAdmin category. Declare Pages rules belong to the Technical category.

5. What is forward chaining and Backward chaining? Explain with examples.

Forward Chaining : Forward chaining provides the automatic calculation of
the property by executing the declarative rule, when any one of the input

property value is changed. For example, if the Area property depends on the Length and Width property, then forward chaining causes the Area property to be recomputed each time either Length, or Width value changes.

BackwordChaining :Backwordchaining provides the automatic calculation of the property by executing the declarative rule, when a value is needed for a property, rather than when inputs change. For example, if the Area property depends on the Length and Width property, then backword chaining causes the Area property to be recomputed each time the Area property is required/ referred

What decides the forward chaining / Backwordchaining ?: The "Target Property Data " field in declare expressions decide this
In the above options,
Whenever inputs change – Forward Chaining
When used, if no value present – Backword Chaining
When used, if Property is missing – Backword Chaining
Whenever used – Backword Chaining

Forward chaining provides the automatic propagation of changes in one property value to cause recomputations of another property's value.

For those Declare Expression rules with Whenever inputs change in the Calculate Value field, the system creates an internal dependency network, identifying which properties have values that may depend on which other properties.

For example, if the Area property depends on the Length and Width property, and the Volume depends on the Area and Depth property, forward chaining causes the Volume property to be recomputed each time Length, or Width, or Depth changes value.

Informally, the term forward chaining can apply to other automatic propagation of changes, such as by Declare Trigger and Declare OnChange rules. When these execute, they may in turn cause additional automatic processing to occur, and so on.

Backward chaining

backward chaining executes Declare Expression rules when a value is needed for a property, rather than when inputs change. For example, a value may be needed because the property appears as a source in a Property-Set method or as an input to a computation.

Backward chaining applies to Declare Expression rules with the Calculate Value field set to one of the following:

- When used if no value present
- When used, if property is missing
- Whenever used

If the formula Area = Length times Width is set up for backward chaining with the Whenever used option, then each time the system accesses the Area property for a value the Declare Expression rule recomputes. This recomputation may occur more often, or less often, than the inputs Length and Width change values.

If the system needs a value for Area through backward chaining, but Length has no value, the chaining may continue using a Declare Expression rule for Length, and so on. Alternatively, the application can prompt a user to enter a value.

6. Activity Types :

The **Activity Type** is a field on the Security tab of the Activity form describing the activity's characteristics. An Activity Type has one of the twelve values listed here.. Five of these types identify activities that you can reference directly in flow rules.

Activity Type	**Description**
Activity	Any activity that is not one of the activity types below.
Assembly	Reserved. Do not use.
Assign	An activity that creates an assignment, an instance of a concrete class derived from the Assign- base class. Referenced in a flow to support an assignment. These activities may be referenced by an assignment shape (or) on a flow rule.
Connect	An activity that calls a Rule-Connect- external system interface. These activities may be referenced by an Integrator shape (or) on a flow rule.
Locate	An activity that finds a locatable page; referenced on when condition, Declare Expression and Constraints rules only.
Load Declarative Page	An activity that adds values to declarative pages. Referenced on Declare Pages rules only.
Notify	An activity that sends correspondence to a work party in a work item. In a flow rule, such activities may be referenced in the Properties panel for an assignment, or explicitly in a Visio notify shape ( ).
OnChange	An activity that starts automatically when another activity step changes certain properties, as defined through a Declare OnChange rule.
Route	An activity that determines which user worklist or workbasket receives an assignment. In a flow rule, such activities may be referenced in the Properties panel for an assignment, or explicitly in a Visio router shape ( ).

Trigger	An activity that starts automatically when an object of a certain class is saved, as defined through a Declare Trigger rule.
Utility	An activity that typically updates a work item and is referenced in a flow, but requires no user interaction. In a flow rule, these activities may be referenced in a utility shape (or) .
Validate	An activity that validates the contents of a rule or data form. (Rarely needed in applications.)

The Assign, Connect, Route, Notify and Utility types identify activities that are directly referenced in flow rules. Typically, activities used in flows have a class derived from the Work- base class as the Applies To portion of their key.

The Activity Type field on the Activity form corresponds to the property Rule-Obj-Property.pyActivityType

7. Pega guardrails?

- Adopt an Iterative Approach
- Establish a Robust Foundation
- Do Nothing That Is Hard
- Limit Custom Java
- Build For Change
- Design Intent-Driven Processes
- Create Easy-To-Read Flows
- Monitor Performance Regularly
- Calculate and Edit Declaratively, Not Procedurally
- Keep Security Object-Oriented Too

8. Diff b/w Obj-Open and Obj-browse?

Obj-Open opens a single instance using the Data tables primary key which maps to the class while browse browses multiple instances of same class

Use the Obj-Browse method to search instances of one class and copy the entire instances, or specified properties, to the clipboard as an array of embedded pages.

Only properties exposed as columns can be used as selection criteria. However, values of properties that are not exposed as columns, including embedded properties, can be returned.

Execution of this method does not trigger declarative rule processing that depends on change tracking. Even when a property retrieved by the Obj-Browse method is involved in a Declare Expression, Declare Constraint or other declarative rule, retrieval does not cause the declarative rule to re-evaluate.

9. Locatable Pages:

A capability called **locatable pages** reduces the need for pages to be explicitly named in the rule. Locatable pages can be used with Constraints rules and Declare Expression rules.

References to properties on a locatable page use the locate keyword in lowercase as a prefix or initial portion of the page name, as in:

```
locateCustomer.OrderAmount
```

At runtime, the system uses backward chaining and an activity with type Locate to find and (if necessary) add the proper page to the clipboard. This eliminates the need for the developer creating the when condition rule to specify the page.

10. How do you expose a property? What is the need of exposing a property?

A Single Value property that is visible as a column in a database table is said to be **exposed**. Aggregate properties, properties within an embedded page, and properties that are not exposed are contained in a specially formatted Storage Stream or BLOB column. Most PegaRULES database tables contain a Storage Stream column named pzPVStream.

Which properties are exposed affects the record selection operations in list view and summary view rules. In many cases, your database administrator can cause a property previously stored only inside the Storage Stream column to become a separate exposed column.

Only top-level Single Value properties can be exposed. If your application needs a column corresponding to an embedded property values, the values can be copied to the top level or exposed indirectly through instances of an Index- class.

Don't confuse an exposed property with an **indexed** database column. For example, the property Work-.pyID is an exposed property in the table pc_work. However, if this table contains several million work items, a list view search for a .pyID value of "W-135" causes an expensive database table scan. Creating database indexes can speed database operations in such situations. Ask your database administrator to create database indexes.

A top-level Single Value property that is not exposed as a column is sometimes called an **unoptimized property**, indicating that its values are not easily available to be referenced in report definition rules. When the Display Unoptimized Properties in Data Explorer checkbox is enabled (on the **Data Access** tab of the Report Definition form), report users can choose to include such properties in reports, as column data or row selection criteria.

You can use any of these tools to expose a top-level Single Value property as a column:

- Modify Schema wizard — > System > Database > Modify Schema
- Property Optimization tool — From the Application Explorer right-click menu
- Database Class Mapping landing page tab — > Data Model > Classes and Properties > Database Class Mappings, for tables in external SQL databases.

To expose embedded properties

Only top-level Single Value properties can be exposed as columns. Two tactics are available when you need a database column that contains the value of an embedded property:

Copy approach — Copy the value to a new top-level property each time the embedded property changes (or each time the instance containing the property is saved). For example, if a list view report needs to select rows based on property pyWorkPage.pxFlow("LoanDisburse").Sheet.Detail(4), you can:

1. Create a new top-level Single Value property to hold a copy of this value.
2. Create a one-step activity to copy the value to the new top-level property, with Activity Type set to Trigger.
3. Create a Declare Trigger rule that calls the activity each time the work item is saved
4. Expose the top-level property.
5. Reference the top-level property in the list view rule.

Declare Index approach — If not one but many or all values of a Value List or Value Group are needed as exposed columns, a Declare Index rule is a better approach.

1. Create a concrete class derived from the Index- base class.
2. Create Single Value properties in the new class to hold values of the embedded values.
3. Create a Declare Index rule with the appropriate embedded Page Context value that copies the embedded values into a new Index- instance.
4. Save the Declare Index rule. It executes immediately, adding and deleting instances of the new class.
5. Expose database columns corresponding to the Index- class.
6. Reference the Index- properties in the list view rule.

11. what is Clipboard?

- **The Process Commander logical layer for storing data between an application and the database is the clipboard. Process Commander keeps XML-based structures called pages in memory, and pages contain property-value pairs. A page is an instance of a class rule, and the page holds the current property values for that instance.**
- **Clipboard Pages:**
- **The Clipboard has a hierarchical structure, consisting of nodes known as pages, most of which have a name and an associated class.**
- **Pages act as buffers or temporary copies of object instances (of that class) that are copied from, or may later be stored into, the PegaRULES database or another database.**
- **The Clipboard contains three broad categories of Top-Level Pages:**
- **User Pages**
- **Declared Pages**

- **System-Managed Pages**

12. What is the use of step page in activities?

Identify the page on which this step is to act. If you leave this field blank, this step acts on the primary page of the activity.

All page names must also appear on the **Pages & Classes** tab.

For most methods and the Java instruction, the step page must already exist. However, the following methods create a page from the Step Page information when it does not exist:

- Property-Set-*
- Connect-*
- Obj-Open
- Obj-Open-By-Handle
- Obj-Refresh-And-Lock
- Page-Change-Class
- Page-Merge-Into
- Page-New
- RDB-Open

13. What is a declare page?

A **declarative page** is a clipboard page created by execution of a declare pages rule (Rule-Declare-Pages rule type).

The name of a declarative page starts with the prefix Declare_. Such pages are created and updated only through operations identified in a Declare Pages rule. The contents of these pages are visible but read-only to requestors.

Benefits

Declarative pages can improve performance and reduce memory requirements **when all or many requestors in an application need to access static information or slowly changing information.**

For example, a declarative page may hold last night's closing U.S. stock prices in a Value Group property indexed by ticker symbol, so that the property reference Declare_Stock.Price("IBM") is the closing price for IBM shared. The first time each evening (after the 4:30 P.M. New York market close) that a requestor attempts to access the page, the system automatically loads the page with the latest end-of-day prices. The page can remain unmodified in memory until the next day's closing.

In other situations, data values are not static but may change infrequently; the system automatically checks the declarative page contents (using a when condition rule) before each property access to see whether a fresh recomputation is needed. For example, a page may list the part numbers or SKU numbers of items that are out-of-stock, extracted from an inventory control system. Recomputation is needed only as often as an out-of-stock condition begins or ends, not each time the inventory changes.

Notes

Declarative pages are not created at system startup; they are created only when first accessed.

Ordinarily, declarative pages do not persist; they are not saved to the PegaRULES database but are reconstructed when next accessed and the previous version is expired.

On the Clipboard tool display, declarative pages appear as a group named Declared Pages.

On the Performance tool full details display, you can review (for your own requestor session) elapsed time and CPU time statistics associated with creating, finding, and accessing declarative pages, and counts of the number of node-level declarative pages your requestor has accessed.

The system maintains node-level declarative pages in a cache.

14. Difference between activity and utility

Utility is specified only in the flow rule.If you want to perform some action without human intervention we need to use utility. Activity should need the human intervention

15. Explain Declarative rules?

Declarative rule

A **declarative rule** is an instance in a class derived from the Rule-Declare- class.

You can establish required relationships among properties in a Declare Expression, Constraints, Declare OnChange or Declare Trigger rule. When the value of a property is involved in any of these declarative rules, the system automatically checks an internal dependency network for other values that are affected and performs other processing as determined by the network. This is known as **forward chaining**.

After the following types of events the system reevaluates most declarative rules:

- When users submit an input form, evaluation occurs at the conclusion of input processing.
- As an activity runs, evaluation occurs during each step, after the method completes but before evaluation of a transition in that step.
- During flow execution, as control advances from one shape on the flow diagram to another.
- At the conclusion of a flow transition, when the work item advances from one shape to another and within connectors (if a relevant property is set).

Index rules operate when a value of any of the properties involved in the rule changes.

Trigger rules operate when an object is saved to the database.

Decision tree rules, decision table rules, and case match rules are evaluated only upon explicit request, and do not use forward chaining.

Limited rule resolution applies to declarative rules, based on the RuleSet, Value, and Available fields. You can use circumstance processing and time-qualified processing with Declare OnChange, Declare Expression, and Declare Constraints rules.

Declarative processing simplifies your application and reduces the number of activities you need to create. The declarative processing facilities support operation of six rule types:

- Constraints rules (Rule-Declare-Constraints rule type)
- Declare Expression rules (Rule-Declare-Expressions rule type)
- Declare Index rules (Rule-Declare-Index rule type)
- Declare OnChange rules (Rule-Declare-OnChange rule type)
- Declare Trigger rules (Rule-Declare-Trigger rule types)
- Declare Pages rules (Rule-Declare-Pages rules type)

On the Application Explorer and Rules Explorer display, four of these rule types belong to the Decision category. Declare Index rules support reporting and belong to the SysAdmin category. Declare Pages rules belong to the Technical category.

A declarative rule describes a computational relationship among property values that is expected to be valid "always" or "often" or "as needed." In practice, declarative rules are evaluated automatically when needed to satisfy certain conditions. These rule types differ in processing to offer you specific meanings of "often" or "as needed."

The primary benefit of declarative processing is that the system, rather than the developer, controls when computations are processed. Relationships between property values can be expressed directly, without the need to create activities and design the application to run the activities often.

Declare Expression rules

A **Declare Expression** rule defines a computational relationship among properties, such as:

```
    Area is computed as Length multiplied by Width
```

After you save a Declare Expression rule, you can be certain that thereafter the value of the Area property, when accessed on the clipboard, is equal to the product of the values in the Length and Width properties. Any change to a Length or Width value, regardless of how the change occurs, causes immediate recomputation of the value of the Area property. This works even if the update to Length happens indirectly.

Constraints rules

A **Constraints** rule records an expected relationship between property values, such as:

```
ActualExpense("December") is never greater than BudgetExpense("December").
```

As with expressions, the system monitors whether the constraint is met each time either value changes. If the constraint becomes false, the system adds a message on the page containing the left-hand property

(ActualExpense(), in this case), marking the page as not valid. Pages that are not valid cannot be saved to the database.

Declare Trigger rules

A **Declare Trigger** rule identifies processing to occur automatically when an instance of a specific class is saved or deleted. (An activity with an Activity Type of Trigger performs the processing.) For example, you can maintain the total month-to-date dollar value of new work items with a Declare Trigger rule. Each time a new object is resolved, the trigger rule executes; the trigger activity runs and updates the total. (Because this processing occurs in the current user requestor, make sure that the design saves the total in a separate object, and that locking ensures that only one user can update the total at a time.)

Declare OnChange rules

A **Declare OnChange** rule causes computation to occur when the value of certain "watched" properties change. This may cause more frequent updating than a Declare Expression rule. A Declare OnChange rule starts an activity that can perform more complex or comprehensive computations than can be defined by expressions.

Declare Index rules

A **Declare Index** rule creates or deletes new objects in concrete classes derived from the Index- base class. Like other declarative rules, the rule operates automatically and as needed.

However, index objects support specific data-access needs rather than computation. Accordingly, Declare Index rules are best designed by database administrators or Process Commander developers with database experience. Unlike the other rule types described here, Declare Index rules are part of the SysAdmin category.

16. Performance Tools

Preflight

- **List each rule in an application that contains a warning message. Warning messages suggest that the rule is at variance with guardrails and other best practices.**
- **Check for browser compatibility of visual elements.**
- **Displays rules with warning messages in the form of bar, pie, summary report.**

Use the Performance page to access three tabs.

- PAL
- Database Trace
- Performance Profiler

PAL tab

Use the PAL tab to understand the system resources consumed by processing of a single requestor session, or the SQL statements sent to the PegaRULES database by the requestor session.

You can interact with the summary page to record statistics. See Performance tab — Using the Summary display. Click the INIT, FULL, or DELTA links to access the Full Details display for that row. This display provides additional statistics from the same snapshot. See Performance tab — Full Details Display.

Click:

- Add Reading — A new row of summary performance data appears. Statistics include input/output, elapsed time in seconds for certain processing, cache performance, locking, and server CPU demand, for the most recent interaction. Optionally, first click elsewhere in the portal window and perform one or several interactions that cause the performance you want to measure. The system retains extensive data for each interaction.
- Add Reading with Clipboard Size — A new row of summary performance data appears, including the size of the clipboard in bytes. Similar to Add Reading, but may add potentially costly processing.
- Reset Data — Initializes all counts and totals to zero.
- Save Data — Lets you save more than 100 statistics and values from each row into a Comma-Separated-Values file (CSV file type) for analysis with Microsoft Excel or other facilities. Column headings correspond to properties in the Code-Pega-PAL class, as presented on the Full Details display. (The value NaN indicates that a statistic was not available.)

Process Commander always accumulates cumulative resource statistics for the Performance tool. Use the landing page to display these statistics, and to identify incremental resources (in the delta rows) consumed by your processing. Because this feature displays existing data, its use does not degrade processing.

Database Trace tab

You can monitor the interactions between Process Commander's server engine and the PegaRULES database or external relational databases, and the operation of the rule cache. Familiarity with SQL is required to interpret the output.

Operating this tool can adversely affect performance, and can produce a large volume of output. Use this tool only in appropriate non-production debugging environments, for short periods.

This tab is available only to users who hold the Code-Pega-.PerformanceTool privilege.

Before starting the Database Trace tool, click the Trace Options button and complete the options form. See Setting Database Trace options. Select only the minimal set of options that are required to aid your debugging. Click OK, then click Close.

Click to start the Database Trace tool. Perform the operations that you want to trace. Click the red square to stop the tool. Trace details appear in the table, summarized by Thread.

Field	Description
User	Operator ID.
Thread	Name of the Thread that generated this row.
Started	Date and time that database tracing started.
Stopped	Date and time that database tracing stopped.
Recorded Time	Elapsed time between start time and stop time.

Click the Save Icon at the right of each row to save the extensive trace details from that Thread to a local text file. Click Start Excel if you want to upload the saved text file and parse the text values into an Excel spreadsheet. After Excel starts, click Get Data and Create Views to import and process the saved file. See Interpreting Database Trace Results.

System-wide database trace

An alternative approach that provides comprehensive tracing of SQL statements sent to the PegaRULES database is the dumpStats parameter in the prconfig.xml file.

To enable this feature:

1. Update the <database> node of the prconfig.xml file to add this element:

```
<entry key="dumpStats" value="true" />
```

2. Stop and restart the application server.

As an alternative to the **prconfig.xml** file, you can use Dynamic System Settings to configure your application.
See How to create or update a prconfig setting.

This setting generates a system-wide database trace file in the ServiceExport directory that can become very large quickly, and can affect system performance. Use this setting only for brief periods, and when a single-requestor DB trace is not suitable.

Performance Profiler tab

Use the Performance Profiler tab to obtain a detailed trace of performance information about the execution of activities, when condition rules, and data transform rules executed by your requestor session. The Profiler traces every execution (in all Threads) of rules of these three types in all RuleSets. Trace records are saved as instances of the Log-Trace-Profiler class.

This tab provides more performance details than the Tracer tool or the Performance tool. However, when enabled, this tool produces extensive output and requires substantial processing overhead. **Disable this tool** as soon as your data collection is complete.

17. Explain Decision rules?

18. Diff b/w Obj-List and Obj-browse?

In Obj Browse search criteria fields need to be exposed whereas in Obj-List search criteria fields need not be exposed.Obj-List is depricatedbecoz of the very reason that it ll retrieve entire blob and then try to filter results based on provided values which affects performance.

19. What is cover and covered objects. What Covers and Folders?

The basic units of work can be simple work objects or work covers or work folders. If a work object is considered as a simple file , then a work cover can be thought of as another file which can contain another embedded file(s) and a work folder as a folder having many files and also folders.

So the basic units of work and their associated work classes can be shown in class structure form as below:

- ***Work-Object-***
- ***Work-Cover-***
- ***Work-Folder-***

- **A cover is a work object that is also a parent to one or a few other related work objects. Typically one work party — such as the customer party — is present in the cover work object and also present in all of the covered work objects associated with it. The covered work objects are the children in a parent-child relationship.**
- **Internally, a cover is a work object in a concrete class derived from the Work-Cover- abstract class.**
- **A cover work object provides a means to coordinate processing of the related work objects. Normally, the system resolves a cover work object once all its "member" covered work objects are resolved. The standard ticket rule Work-Cover-.AllCoveredResolved alerts a cover flow that all the member work objects have been resolved.**
- **Locking a covered work object normally also locks the cover work object. This is desirable as the cover work object may contain totals, balances, counts or other derived information that require single-threaded access. Normally, the cover work object is added to the clipboard using a page named pyCoverPage; the covered work object is on a page named pyWorkPage.**
- **The covered work objects may be of various work types. However, the work type of the cover and the work type of the covered objects must all belong to the same work pool.**

- By convention the default Work Object ID of a cover starts with a C-, as opposed to that of a Standard Work Object, which starts with W-.
- A cover is a work object of a concrete class derived from the Work-Cover- abstract class.
- Covers cannot be nested. A cover cannot cover another cover; likewise, a cover cannot be covered.
- There is a one-to-many relationship between the cover and its covered work objects. Covered work objects can only belong to one cover. A cover should not contain more than 20 covered work objects.
- Covers and the work objects they contain must belong to the same application class group.
- A cover and its covered work objects are related to each other by reference using property settings.
- Locking a covered work object normally locks the cover work object as well. This is desirable as the cover work object may contain totals, balances, counts, or other derived information that needs to be carefully managed. Note: This behavior can be overridden in a standardized way.

❖ Cover processing is managed using tickets.

Folders

- By convention, the work object ID of folders has the format F-.
- A folder is a work object of a concrete class derived from the Work-Folder- abstract class.
- There is a many-to-many relationship between folders and work objects. The same work object can be added to multiple folders.
- Folders can be nested and can contain folders, covers, and work objects in any combination; but remember covers cannot contain covers.

20. What is the major difference between the Cover and Folder

- **There is a many-to-many relationship between folders and work objects. The same work object can be added to multiple folders.**
- **No built in processing or locking**
- **Usually used to relate similar work objects, such as customer requests by geography, or with a common attribute, such as vendor, merchant, etc.**
- **Typically, BPM features such as service levels and urgency are not used for folders**
- **Need to associate work objects in the database for reporting purposes where a work object can belong to more than one 'collection'**
- **Need to support nested collections**

Covers and folders are two built-in facilities that allow your application to support collections of work objects. In contrast to covers:

- One work object may be associated with multiple folders, but only with one cover.
- Members of a folder can belong to different work types, which need not belong all in a single work pool.
- The relationships between folder work objects and their contents may be many-to-many.

Example

To review or work with a flow rule that creates a folder work object, examine or test the standard flow rule PegaSample-Folder.NewWork.

As a business example, consider a purchasing application where a basic work object is a line item on a single purchase order:

- The purchase order corresponds to a cover object. Every line item belongs to one purchase order.
- Folders organize open purchase order line items by our part number and also by the vendor's part number. A folder work object (in Work-Folder-OurPartNum work type) may be associated with several line items, and one line item can simultaneously belong to a Work-Folder-OurPartNum work object and a Work-Folder-TheirPartNum work object.

Standard rules

These standard rules support folders:

- Work-.AddToFolder flow action — Add a work object to a folder
- Work-.RemoveFromFolder flow actions — Remove a work object from a folder
- Work-Folder-.Review harness — Show folder work object, includes the View Folder Contents button ( ) to access contents
- Work-Folder-.Perform harness — Update folder work object and contents
- Work-Folder-.ReviewforExplore harness — Split-screen presentation
- Work-Folder-.FolderContents section — Show folder contents
- Work-Folder-.NewWork flow — Create a folder

Notes

A folder work can be a member of other folder work objects.

Instances of the Link-Folder class associate of work objects with folders.

21. **Difference between obj-list, rdb-list?**
22. **Difference between obj-open and obj-open-by-handle**
23. **Diff b/w Obj-Method and RDB-method?**
24. **Differentiate Obj-Open and Rdb-Open methods**
25. **Differentiate Obj-Browse and Obj-List-View**
26. **Differentiate Obj-List-View and Obj-Summary-View**
27. **What is indexing in prpc?**
28. **What is default session out time?/ For how much time a lock is held on a work object if it is browser is closed intermittently?**
29. **Class group? Mapping?**
30. **How to map a class to a table?**
31. **Spin off/Split for Each/Split join?**
32. **Different Activity methods?**
33. **Ticket usage and related activity methods?**
34. **SLA?**
35. **How do you handle exception handling?**
36. **What is the best use of Declare pages.**
37. **How to trace an agent?**

38. **What are services and connectors?**
39. **With trace open rule,when you are running an activity stand alone,you could not see any steps in tracer?What could be the reason?**
40. **How to make a rule reusable component?**
41. **Where can you see the local and param values?**
42. **Diff between spin off and split for each?**
43. **Access role?**
44. **What are the different inheritances in pega?**
45. **What is the importance of directed inheritance?**
46. **Which inheritance will have preference?**
47. **Which inheritance is mandatory of these the two?**
48. **Direct and Pattern Inheritance**
49. **What is rule resolution and what parameter defines this resolution algorithm?**
50. **What is circumstance and different types?**
51. **What is abstract class and concrete class?**
52. **Prepare well about all the flow shapes in prpc, they covered almost all the shapes?**
53. **What is local action and what is connector action?**
54. **What is screen flow?**
55. **Explain about bulk processing?**
56. **Question related to repeating layout using onchange client event?**
57. **What is difference between list view and summary view reports?**
58. **What is the option available on correspondence rule form?**
59. **User needs to display the message on screen for the first time and for next time access to that screen should not display that message?**
60. **Question related to exposing properties related to reporting requirement?**
61. **What are the different types of activities?**
62. **What is agent management?**
63. **What are different types of SLAs?**

64. **What is difference between decision table and decision tree and when user needs to opt for decision table or decision tree?**
65. **What is fork and when user needs to use fork?**
66. **What is sub flow and how control will work when using subflows?**
67. **How to make any rule as a favorite to your manager**
68. **Click on favourite icon on the rule to delegate the rule specify the access group to whom you wantr to delegate the rule.**
69. **Where can i see the paramater values in the clipboard (values ..) i am passing one activity to other.**
70. **you cannot see the parameter values on the clipboard.You can see them while tracing**
71. **3. How to import rules using pzinskey**
72. **Performance of our work in the pega is measured using?**
73. **How to connect to different pega applications?**
74. **How to store the instance of the class in a specific database**
75. **How to see values of the local variables of the activity.**
76. **how can i store the instance of the class in the data base**
77. **default data table where the instance of the class are store (how it will search) pc_work**
78. **In Routing activity what is the default property used to route the object**
79. **In routing activity if i use workbasket name instead of work list name .. when can i know it is wrong (run time, compile time)**
80. **Notify**
81. **Ticket: explain any scenario u used**
82. **Table used for add note**
83. **19. Default activity used to create work object**
84. **20. Different type of flows. Explain in scenario based where u used and worked**
85. **Covers and folders (question will be like scenario base. like I have one object A in it 10 test case objects .. if 10 test case are passed ..A should close how can I achieve this functionality.**
86. **work object ID.. how to create.. activitesued to create, or methods Work ID:**
87. **how to send multiple correspondences at a time**
88. **How to call an Activity from Java Script?**

89. how to end the workobject in the activity (method used to kill the work object)

90. how to call an activity from the java, java script

91. How to pass parameters to the activity using the java, JavaScript?

92. How can I pass page as the parameter to the activity using java, JavaScript?

93. How to call an Activity from Java step?

94. How to get a property value from clipboard using Java step?

95. How to restrict the harness, section to particular user

List different functions used to call an activity from java script.

How a user's ruleset list is formed (thelogic)?

96. How to connect external java application without using connect-java

97. Spinoff // split join explain

98. Privileges usage...

99. Decision / fork usage... Scenarios Decision:

100. . How do you connect to a soap service which is secure (https).

101. What is a stateful and stateless Soap service in Prpc.

102. If there are 2 rulesets, ruleset r1 is prerequisite of r2. How can we call an activity present in r2 from

103. activity present in r1.

104. Is the out parameter mandatory? Should we specify the out parameter to receive any output value of an activity

105. If there are 2 rulesets, ruleset r1 is prerequisite of r2. How can we call an activity present in r2 from activity present in r1. We cannot call an activity present in a child ruleset from an activity present in its prerequisite ruleset. Here...

106. How to Call a db stored procedure from Pega

107. We can query database tables using Rdb-list, Rdb-browse, Rdb-open, Obj-Open, Obj-Browse, Obj-list. Well are these built-in Pega activities successful in executing stored procedures? Yes ofcourse we can execute stored procedures from Pega. Use...

108. How do you connect to a soap service which is secure (https).

109. What is a stateful and stateless Soap service in Prpc.

110. If there are 2 rulesets, ruleset r1 is prerequisite of r2. How can we call an activity present in r2 from activity...

111. HTML streams are subject to rule resolution. Is this statement true or false?

112. PegaRULES Process Commander standard harness activities and starting HTML streams are meant to work as-is, with links to built-in processing capabilities. For...

113. How to make any rule as a favorite to your manager

114. Where can i see the paramater values in the clipboard (values ..) i am passing one activity to other .

115. How to import rules using pzinskey

116. Difference between activity and utility 5....

117. What is the Flow Action? Explain about the FlowAction?

118. What is the Activity?

119. What is the Model?

120. What is the Harness? Section?

121. What are the fields in the properties panel of an assignment shape? Where can we call the activities in a flow action? What is Class structure of your Project? Explain about the project Flow? What is the Rule availability? What is the Final...

122. Difference between Java and Pega Guardrails of Pega What do you mean by Build for Change Difference between page and pagelist why we use connect-soap and can we use it to connect external database why we use connect-sql how many...

123. What is SMA? Differentiate Obj-Open VsObj-Browse How do you handle exceptions Differentiate the usage of Assignment Shape and Router shape Where do you define default values What is the primary key of pc_assign_worklist This is how People...

124. What is the difference between Page-Validate and Property-Validate methods?

125. Page-Validate method is used to validate all the properties present on a page. If a page has embedded pages, this method works recursively to validate all the properties. This method consumes lot of system resources and takes more time. If you want to validate specific properties use Obj-Validate method with Rule-Obj-Validate rule.

126. Property-Validate method is used to impose restrictions on a property value. Use Edit validate rule along with Property-Validate method to impose restrictions. You can validate multiple properties using Property-Validate method.

127. 2.What is difference between Edit validate and Edit Input rules?

128. Edit Validate : Use edit validate rule to validate the property value using java code. Edit validate rules can be used property-validate, Rule-Obj-Validate and Property rules.

129. **Edit Input : Edit input rules converts user entered data into required format. For example is the user enters date MM/DD/YYYY format, edit input rule coverts this date into DD-MMM-YYYY (required format). Again we need to write java code for this transformation.**

130. **3.Where assignments will be stored in pega rules database?**

131. **Work List related assignments are stored in pc_assign_worklist.**

132. **Work basket related assignments are stored in pc_assign_workbasket.**

133. **4.Where work objects will be stored ?**

134. **Work Objects are stored in pc_work table by default. however if you want to store the work objects in a user created table, follow the below mentioned steps.**

135. **¦Create a schema similar to pc_work table. (The best thing is to copy the pc_work schema and modify the table name and constraints name if any)**

136. **¦Change the class group mapping (Data-Admin-DB-Table) to the newly created table.**

137. **5.If I have 3 different work objects in my application, how to store them in three different tables?**

138. **Open/Create the Data-Admin-DB-Table instance for each class and mention the table name. By doing this the individual work objects will be stored in the new table you mentioned in the Data-Admin-DB-Table instance. This is a best practice if there too many object instances for each class.**

139. **6.What is StepStatusGood, StepStatusFail rules?**

140. **StepStatusGood is a when condition defined in @baseclass, this when rule checks whether the value of pxMethodStatus property is "Good".**

141. **StepStatusFail is a when condition defined in @baseclass, this when rule checks whether the value of pxMethodStatus property is "Fail".**

142. **¦Difference between Java and Pega**

143. **¦What do you mean by Build for Change**

144. **¦Difference between page and pagelist**

145. **¦why we use connect-soap and can we use it to connect external database**

146. **¦why we use connect-sql**

147. **¦how many shapes you know in pega.**

148. **¦what do you mean by calculate and edit declaratively not procedurally**

149. **¦what are tickets give scenario where you used tickets**

150. **¦What are the 6 R's**

151. **What is SMA?**
152. **2.Differentiate Obj-Open VsObj-Browse**
153. **3.How do you handle exceptions**
154. **4.Differentiate the usage of Assignment Shape and Router shape**
155. **Where do you define default values**
156. **What is the primary key of pc_assign_worklist**
157. **Commonly asked questions in Pega for a beginner or a year experienced.**
158. **What is a class group?**
159. **5.Which activity do we use in Obj-List-View to customize the search results**
160. **6.What is the use of pyDefault model?**
161. **7.Differentiate Decision table, Decision tree, Map value and Map value pair**
162. **8.Difference between forke and decision shape**
163. **How do u debug the harness?**
164. **Is it possible to add a section with in a section?**
165. **Is it possible to add HTML form in a section?**
166. **Is it possible to add HTML rule in a HTML rule?**
167. **Where workbasket and work-list is used?**
168. **How to map work group in a table?**
169. **How to map Access-group?**
170. **About Rules Inspector?**
171. **Where routing flow is used and how it works?**
172. **About list view?**
173. **Describe Property-set method?**
174. **About HTML Fragments?**
175. **Diff b/w value-list and pagelist?**
176. **How do u know is SLA working or not?**
177. **Tell me the class structure in ur project?**
178. **What is flow-action?**

179. **How u create work object?**

180. **Why u need class group?**

181. **What is work pool?**

182. **What is work basket?**

183. **Can u explain me the process for connecting with external database(step wise)?**

184. **What is SLA and how do u know whether it is working or nor?**

185. **How many types of SLA’s are there?**

186. **Can we use obj-methods for external databases?**

187. **Explain Locking system in PRPc?**

188. **How can u validate a property?**

189. **Diff b/w Obj-validate & Property-validate?**

190. **What is the agents.**

191. **How to trace an Agent.**

192. **Have you used spin-off shapes in your application, if so what is the case.**

193. **What are the type of log files we have**

194. **Do you know the concept of validation rules?**

195. **How do you a log a message in PRPC?**

196. **Tell me about the locking concept in PRPC.**

197. **What is DWA (Direct Web Access.**

198. **What is the diff b/n Decision Table and decision tree**

199. **How do u connect to the Web service**

200. **What is xml file name created during connection of web-service**

201. **What is order of execution in Decision Table and decision tree**

202. **Rule resolution algorithm**

203. **Decision table and Decision Tree**

204. **What are the issues faced in Upgrading from 5.3 to 5.5**

205. **How do you purge the work objects?**

206. **What is the role of the master agent**

207. **Do you have any idea about the AES?**

208. **Suppose I need to access the Google service, how can I achieve in pega.**

209. **What are the rules which are not rule resolved?**

210. **What is the actual usage of the blocked, with drawn, NO. When you use it?**

211. **What is the usage of Property-Alias rule**

212. **What are the various ways to restrict user to edit the rules.**

213. **How do you display an image dynamically in alist view rules based different types of urgency.**

214. **How do you build a new Rule-Form**

215. **What is DCO? In how many ways you can create a new PRPC application.**

216. **What is a screen-flow.Difference between screen flow and process flow?**

217. **How do you provide SLA for the work-object? A : Model**

218. **In Connect Sql rule stored procedures are written in which tab?**

219. **Save,Browse,Deleteetc tabs present in Sqlrule.Which tab will be called when?**

220. **What is the difference between work-object and work-type?**

221. **What is the difference between Decision tree and Decision Table?**

222. **If you have work experience on Soap Service.What are the rules required for Soap Service?**

223. **In which tab WSDL file will be generated in the Service Package?**

224. **About the debugging tools like Clipboard,Tracer etc.**

225. **Explain about dynamic select.**

226. **How to use existing css in PEGA?**

227. **SOAP integration using PEGA --------- Techincal/Functional**

228. **Harness,HTML,Section rules----------- Technical**

229. **Agents--------------------------------Technical**

230. **SMART BUILD methodology**

231. **Reports creation--------------List Views/Summary Views/Charts/Graphs**

232. **Rule Resolution-------------Advantages/Disadvantages**

233. **CACHE mechanism**
234. **Performance Evaluation----------Using PAL, SMA(System Mgmt App)**
235. **prconfig,prloggingxmls -----------Idea on configuration**
236. **Pega application deployment on various servers like DEV/QA/PROD**
237. **Differences between V5.x and V4.2**
238. **Migration of Rules**
239. **Test cases generation using V5.4**
240. **Unit testing from developer perspective**
241. **Security**
242. **Accessgroups,portal,operatorsprofile,workbaskets,worklists,workmanager,classgroup**
243. **Work-, Embed- and Data- classes**
244. **Roles, Privileges-------Creation and Manage**
245. **Pega DB schema--------pc_work,pr_other,pc_linketc**
246. **Application Accelrator, Rules Inspector**
247. **Difference between Java and Pega**
248. **What do you mean by Build for Change**
249. **Difference between page and pagelist**
250. **why we use connect-soap and can we use it to connect external database**
251. **why we use connect-sql**
252. **what do you mean by calculate and edit declaratively not procedurally**
253. **what are tickets give scenario where you used tickets**
254. **What are the 6 R's**
255. **what is the work object?**
256. **what is The Class structure?**
257. **how to create Class , class group?**
258. **Access group?How to create Access group?**
259. **5)what is the Harrness ?Section?**
260. **6)what is FlowAction?**

261. **9)what is the Sma?**
262. **10)what is Ticket?**
263. **1)What is Access Group?**
264. **2)What is different types of inheritence?**
265. **3)What is the diifference between Abstract Class and Concrete class**
266. **4)What is work party?**
267. **5)What is a Product?**
268. **9)What is Ticket?**
269. **12)What is pyWorkPage?**
270. **13)What is pz* properties?**
271. **14)What is py* properties?**
272. **19)How do we use Tracer? Trace by rule**
273. **21)What is BPM?**
274. **22)What is difference between BRE and BPM?**

UpdateClaimWithOutreachID