**ASSIGNMENT 4**

**Service Package:**

A service package is a name that groups one or more service rules that are designed to be developed, tested, and deployed together. For some service rule types, this corresponds to a package of Java classes.

To define a service package, create an instance of the Data-Admin-ServicePackage class. Then use this name as the first key part of the service rules.

**SOAP Service:**

SOAP is an XML based protocol used to exchange information in a disturbed and decentralized environment. Services implemented using SOAP is often called as web services.

SOAP service is an instance of Rule-Service-SOAP class.

SOAP services generally process service requests synchronously. That is, they immediately perform their requested processing and return a configured response while the calling application waits. However, you can configure SOAP services to process service requests asynchronously, which means the service queues the request for asynchronous execution and the calling application calls back later for the results

**Service Activity:**

A service activity is an activity rule that is referenced by a service rule. At runtime, the activity operates in a background requestor to perform processing and return results to the client or calling external system.

Service activities are referenced on the service rule form, typically on a form tab named Service. For example, the Service Activity field on the Service tab of the Service SOAP form identifies the second key part of an activity. The system uses information in another field of the form to determine the Applies To class of the activity.

Services operate as a BATCH requestor type and accordingly can't use methods that depend on interactive HTTP processing and human responses, such as the Show-HTML method.

On the Security tab of a service activity, select the May Start? check box to allow the service to start the activity. For maximum security, also select the Authenticate? check box unless the service runs as a guest, without authentication.

Service activity, being called from Service-SOAP/REST rules.

**Service request processor:**

Service request processors, instances of *Data-Admin-RequestProcessor-Service*, specify the following for the services they support:

* The queue class to use – that is, into which queue should the request be saved. You can either use the standard service queue class (*System-Queue-ExecutionRequest-Service-Default*) or create a separate queue class for the service requests processed by that service. If you create one, it must extend the *System-Queue-ExecutionRequest-Service* class so the *ProcessQueue* agent can find and process the requests in the queue.
* How many times to run requests if they fail the first time.
* Whether or not to store the results of successful and/or failed service requests with the queue items and to leave queue items in the queue. If the external application calls back for results, Process Commander should keep the service request in the queue until after the results are retrieved.

You specify which service request processor to use on the Service tab of the service rule. The first key to a service request processor is the name of a service package and a service request processor must belong to same service package as the service rules.

**Queues:**

The Agent queue infrastructure is to store service requests as persistent objects in a queue represented by the standard queue class named System-Queue-ExecutionRequest-Service-Default. When a service queues a service request, the queued item contains all the service request content: the activity, the service class, the primary and parameter pages, the request input, the operator ID to use if it needs to run as an authenticated user, and so on.

**ProcessQueue agent from the Pega-IntSvcs Agents rule runs**

By default, the *ProcessQueue* agent from the *Pega-IntSvcs* Agents rule runs every two minutes. When *ProcessQueue* runs, it processes the item queued by UpdateWorkObject along with all the other scheduled queue items from the queues it monitors.

*ProcessQueue* does the following for the queue items with a status of Scheduled:

* Obtains a lock for the queued service request
* Runs the service request
* If it successfully runs the request, it changes the status of the item to “Success” and stores the results in the queue with the service request queue item.
* If the request fails, the agent increments the number of failed attempts for the item and then checks the MaxAttempts setting in the service request definition. If MaxAttempts is set to more than one attempt, it keeps the status of the queue item set to “Scheduled.” The next time the agent runs it tries to execute the service request again. If there are no more attempts left and the service request continues to fail, the agent changes the status of the queue item to “Broken-Process.”

**Mentioning queuing parameters in requestor:**

The Queue instruction uses no parameters itself. Supply values for the parameters (if any) of the activity.

Parameters given to Activity (entered in the Method field) as:

Type Queue followed by one space and the Activity Name (second key part) of the activity to execute.

Optionally, you can identify the Applies To key part of the called activity explicitly, using the format:

Queue Data-Admin-Operator-ID.Analyze

Queue MyCo-Finance-Work-Form990.Validate

The explicit class format causes rule resolution to bypass all class hierarchy searches — the activity must be present in the specific class identified before the dot. (Other aspects of rule resolution operate normally.) Because use of explicit class names can introduce inflexibility, complicate debugging, and reduce opportunities for rule reuse, avoid this format whenever possible. When you save an activity, Queue steps that use this format cause a rule warning.

Name: Ganga Devi Bandi

SAP ID: 51749543