Lesson 5 The Job Lifecycle

5.1 Requirements

Examine the PolicyChange process in the base application.

5.2 Exercise

Investigation

1. **Find the components to examine:**
   a) Log in to PolicyCenter as Alice Applegate (aapplegate/gw).
   b) Select any policy and start a Policy Change transaction.
   c) On the very first page, press `Alt+Shift+I` to get information about the PCF files.
   d) What is the name of this page?
   e) Close the Location Info window and click Next. On the Offerings screen, press `Alt+Shift+I` to get information about the PCF files.
   f) What is the name of the wizard?
   g) What is the name of the button set?
   h) Close the Location Info window. How many buttons are visible in the UI?
   i) Withdraw the transaction. Note the message that appears.
2. **Locate these components in Studio:**
   a) If necessary, start Guidewire PolicyCenter Studio.
   b) Navigate to `configuration/config/Page Configuration/pcf/job/policychange`
3. **Examine the `StartPolicyChange` page.**
   a) How many entry points does this page have, and what variable must be passed to it?
   b) Where and how does the `effectiveDate` variable get set? (Two answers.)
   c) What does the `Next` button do?
   d) What is its visibility property?
4. **Examine the buttons in the Policy Change toolbar button set:**
   a) What is the PCF file name or the button set?
   b) Open the button set. How many buttons are visible?
   c) Note that two buttons have the same button text. What is the difference between these two buttons? Will they ever appear at the same time in the UI?
   d) What is the action for the Issue Policy button?
   e) Drill down on this action. Where does it reside (i.e., what object is it associated with)?
   f) When a user clicks the Withdraw Transaction button, what causes the confirmation message to appear? (Hint: Two properties are involved.)
5. **Examine how the base application works with the user to handle preemptions:**
   a) What is the visibility property of the Handle Preemption button?
   b) Drill down on the "`can`" function. What conditions does it check for?
   c) Return to the toolbar. What is the action for the `Handle Preemption` button?
   d) Drill down on the popup. What is the action for the `Apply All Changes` button?
   e) Drill down on this action. Where does the script reside?

f) In the `applyChanges` function, locate the `handlePreemptions` function. Where does it reside and what does it do?

Stop

5.3 Solutions

Solution

1. Find the components to examine:

a) Log in to PolicyCenter as Alice Applegate (aapplegate/gw).

b) Select any policy and start a Policy Change transaction.

c) On the very first page, press `Alt+Shift+I` to get information about the PCF files.

d) What is the name of this page?

`StartPolicyChange.pcf`

e) Close the Location Info window and click Next. On the Offerings screen, press `Alt+Shift+I` to get information about the PCF files.

f) What is the name of the wizard?

`PolicyChangeWizard.pcf`

g) What is the name of the button set?

`JobWizardTooldbarButtonSet.PolicyChange.pcf`

h) Close the Location Info window. How many buttons are visible in the UI?

Depends on policy – typically, 4 or 5

i) Withdraw the transaction. Note the message that appears.

2. Locate these components in Studio:

a) If necessary, start Guidewire PolicyCenter Studio.

b) Navigate to `configuration/config/Page Configuration/pcf/job/policychange`

3. Examine the `StartPolicyChange.pfc file`.

a) How many entry points does this page have, and what variable must be passed to it?

One entry point, which requires a PolicyPeriod variable

b) Where and how does the `effectiveDate` variable get set? (Two answers.)

i. *This variable is set in the Variables tab. The initial value is set to the* `EditEffectDate` *of the latest bound policy period. (For example, if the* `EditEffectiveDate` *of the first issued PolicyChange is 3 months after the Policy's effective date, then the second PolicyChange will have the initial* `EditEffectiveDate` *as 3 months after the Policy's effective date.)*

ii. *After that the user can change it using the* `DateTimeInput` *widget "Effective Date". If the date is outside of the policy term's date range, an error message is displayed.*

c) What does the `Next` button do?

- *Starts a new PolicyChange job and commits the database transaction*
- *calls the* `applyEffectiveTimePluginForPolicyChange` *function*
- *then navigates to the PolicyChange Wizard*

d) What is its visibility property?

It is set in the `canVisit` *property with permissions to view the policy period and to create policy change.*

4. Examine the buttons in the Policy Change toolbar button set:

a) What is the PCF file name or the button set?

`JobWizardToolbarButtonSet.PolicyChange.pcf`

b) Open the button set. How many buttons are visible?

10

c) Note that two buttons have the same button text. What is the difference between these two buttons? Will they ever appear at the same time in the UI?

There are two buttons for "Apply change to renewal".

One applies changes to an unbound renewal, the other to a bound renewal. It is visibility is controlled by conditions in the visible property.

They will not appear at the same time in the UI.

d) What is the action for the Issue Policy button?

```
jobWizardHelper.requestIssueJob(policyPeriod)
```

e) Drill down on this action. Where does it reside (i.e., what object is it associated with)?

It resides in the `JobWizardHelperEnhancement.gsx` *entity.*

f) When a user clicks the Withdraw Transaction button, what causes the confirmation message to appear? (Hint: Two properties are involved.)

1) A display key listed in the `confirmMessage` *property.*

2) The `showConfirmMessage` *property is set to true.*

5. Examine how the base application works with the user to handle preemptions:

a) What is the visibility property of the Handle Preemption button?

`policyChangeProcess.canHandlePreemptions().Okay`

b) Drill down on the "`can`" function. What conditions does it check for?

`HasUnhandledPreemptions` *and* `BranchNotLocked`

c) Return to the toolbar. What is the action for the `Handle Preemption` button?

`HandlePreemptionPopup.push(wizard, jobWizardHelper, policyPeriod, true)`

d) Drill down on the popup. What is the action for the `Apply All Changes` button?

`applyChanges(wizard, jobWizardHelper, policyPeriod, CurrentLocation)`

e) Drill down on this action. Where does the script reside?

`HandlePreemptionPopupUIHelper.gs`

f) In the `applyChanges` function, locate the `handlePreemptions` function. Where does it reside and what does it do?

It is a method of `JobProcess.gs` *and (according to the comments) it resolves unhandled preemptions and returns the new policy period.*

5.4 References

5.4.1 The job lifecycle

1. **Below are some of the special job components:**

- **<Jobtype> entities**: contain required data for a policy transaction. Job is the parent entity and subtypes include Submission, PolicyChange, Cancellation, Reinstatement, Renewal, Rewrite, Audit, Reinsurance, etc.
- **Branches**: are sets of objects related to one `PolicyPeriod` object. A branch is one version of the Policy.
- **<Jobtype>Process** classes: contain methods required for processing the job. `JobProcess` is the parent class. Sub classes include `SubmissionProcess`, `PolicyChangeProcess`, `CancellationProcess`, `RenewalProcess`, `AuditProcess`, etc. They are located in `configuration/gsrc/gw/job`.
- **<Jobtype>Wizards**: contain screen, tools and buttons that work with the user in moving through the life cycle of the job.
- **Job Wizard Helper**: is a java class that provides functionality for job wizards.
- **Job Wizard Button Set**: contains relevant buttons for a job type in the job wizard.

2. **Jobs can be started in three ways:**
   - By users through the Actions menu
   - By external systems through APIs
   - Through PolicyCenter batch processes
3. **When the job is started in the user interface, the <Jobtype> object is instantiated by the screen that calls the job wizard.**
4. **A series of job conditions can be checked before the business logic is performed.**

   Typically, they are the can methods in the job process classes. The job wizard button visibility is also based on these can methods, which return a `JobConditions` object. The `JobConditions` object contains a string builder called `_message`. If the `_message` string remains empty after all the error condition checks, the derived property `Okay` returns true, otherwise false.

5. **The business logic can be executed, for example, edit, request quote, issue, etc. from the wizard buttons, the job process methods and workflows. The workflow maybe started in the job process methods as well.**
6. **When the job is bound (or issued), withdrawn, discarded, its life ends. The job object remains in the database and can be retrieved for reference. But it cannot be edited**