GUIDEWIRE

## BillingCenter 10 Configuration: Essentials

# Student Workbook

*Labs and Tutorials*


*Document Published: 2020-07-23*

# Table of Contents

# Introduction

Welcome to the Guidewire BillingCenter 10.0 Configuration Essentials course.

The Student Workbook will lead you through the course labs. The lesson numbers correspond to the lesson numbers in your training. Complete the assigned labs to the best of your ability.

Lesson 1

# Configuring Assignment

## 1.1 Prerequisites

For this lab, use BillingCenter 10.0 Education Installer, Guidewire Studio, and a supported web browser. http://localhost:8580/bc/BillingCenter.do is the default URL for BillingCenter. To test configuration changes, log in to BillingCenter as Super User. The login/password for Super User is su/gw.

## 1.2 Configuring activity assignment

The customer wants all notification activities with Urgent priority to be assigned to the Central Billing group and distributed evenly amongst active users in that group.

### 1.2.1 Requirements

**Spec 1** Notification activity with Urgent priority assigned to the Central Billing group.

**Spec 2** Notification activity with Urgent priority assigned to users in the Central Billing group by round robin.

**Spec 3** Log all assignment activity at debug level.

### 1.2.2 Tasks

1. **Create a new AssignmentUtil_Ext class**
   a) Create the new class in training.bc.utilities package.
   b) Create a findGroup method that finds a group object given a group name.
2. **Create a new GlobalActivityAssignment rule called GAAR1000 – Notification Activity that assigns all urgent notification activities to the Central Billing group.**
3. **This assignment rule must execute before other assignment rules in the GlobalActivityAssignment rule set.**
4. **Restart the server because a new rule was created.**

### 1.2.3 Testing procedure

1. **Write down all users in the Central Billing group.**
   a) Administration Tab: Groups
2. **Create a new notification activity.**

a) Desktop Tab: Actions → New Assigned Activity → Reminder → notification

| Field | Value |
|---|---|
| Subject | ART |
| Description | Assignment Rule Test |
| Priority | Urgent |
| Assigned to | Use automated assignment |

**3. Verify activity was assigned a user in the Central Billing group.**

a) Search Tab: Activities

## 1.2.4 Solution

**1. Create a new AssignmentUtil_Ext class:**

a) Create the new class in training.bc.utilities package.

b) Create a findGroup method that finds a group object given a group name.

```
package training.bc.utilities

uses gw.api.database.Query
uses gw.api.database.Relop

class AssignmentUtil_Ext {
  /**
   * Find group given group name
   * @Param("groupName", "The name of the group")
   * @Returns ("The Group object based on the group name of the group)
   */
   public static function findGroup(groupName : String) : Group {
     var groupQuery = Query.make(Group).compare(Group#Name, Relop.Equals, groupName)
     var resultObj = groupQuery.select().AtMostOneRow
     return resultObj
   }
}
```

**2. Create a new GlobalActivityAssignment rule called GAAR1000 – Notification Activity that assigns all notification activities to Central Billing.**

```
USES:
uses gw.api.system.BCLoggerCategory
uses training.bc.utilities.AssignmentUtil_Ext

CONDITION (activity : entity.Activity):
    return
        activity.ActivityPattern.Code == "notification" and
        activity.Priority == Priority.TC_URGENT
ACTION (activity : entity.Activity, actions : gw.rules.Action):
    var group = AssignmentUtil_Ext.findGroup("Central Billing")
    if (activity.assignGroup(group)) {
      BCLoggerCategory.RULES.debug("GAAR1000 was successful")
      actions.exit()
    } else {
      BCLoggerCategory.RULES.debug("GAAR1000 was not successful")
    }
END
```

**3. This assignment rule must execute before other assignment rules in the GlobalActivityAssignment rule set.**

Drag and drop the new rule at the top of any other assignment rules in this rule set.

4. **Restart the server because a new rule was created.**

Lesson 2

# Configuring Escalation

## 2.1 Create an escalation activity

The customer wants to create an escalation activity for trouble tickets that have been escalated.

### 2.1.1 Requirements

**Spec 1** Create an escalation activity for escalated trouble tickets.

**Spec 2** The escalation activity should not be created if the trouble ticket is already escalated.

**Spec 3** The escalation activity should not be created if an escalation activity already exists for the escalated trouble ticket.

**Spec 4** The default due date for the escalation activity is 10 days.

**Spec 5** The default escalation date for the escalation activity is 30 days.

**Spec 6** Set the escalated activity to High priority.

**Spec 7** The subject of the escalation activity should say "Trouble ticket <Ticket #> escalated."

**Spec 8** The description of the escalation activity should say "Trouble ticket <Ticket #> assigned to <User Name/Group Name> has been escalated."

**Spec 9** Assign the escalation activity to the group supervisor for the escalated trouble ticket.

**Spec 10** Associate the escalation activity to the escalated trouble ticket.

### 2.1.2 Tasks

1. **Create an Escalation activity pattern based on the specifications.**
2. **Create a trouble ticket preupdate rule that creates the escalation activity.**
   a) During the preupdate rule execution, the troubleTicket.Escalated property is equal to true due to trouble ticket escalation batch process.
3. **Restart the server because a new rule was created.**

### 2.1.3 Testing procedure

1. **Create an account.**
   a) QuickJump: Run Account

2. **Create a new trouble ticket and set the due date and escalation date to BillingCenter system date.**
   a) Complete the Trouble Ticket wizard.
   b) Assign the trouble ticket to Aaron Applegate in the Personal Lines group.
3. **Open the trouble ticket and make sure an escalation activity is not created.**
4. **Edit the trouble ticket and make any update. Make sure an escalation activity is not created.**
5. **Run trouble ticket escalation batch process**
   a) *Open Server Tools (ALT-SHIFT-T)*
   b) *Run the Trouble Ticket Escalation process from Server Tools → Batch Process Info*
6. **Verify the escalation activity has been created correctly and assigned to the supervisor of the Personal Lines group, Bruce Baker.**
7. **Run the trouble ticket escalation batch process again to make sure a duplicate escalation activity is not created.**
8. **Edit the trouble ticket and make any update. Make sure a duplicate escalation activity is not created.**

## 2.1.4 Solution

1. **Create an Escalation activity pattern based on the specifications**
   a) Navigate to Administration → Business Settings → Activity Patterns
   b) On the Activity Patterns screen, click the New Activity Patterns button
   c) In the subject field enter, “Escalation Activity”.
   d) In the code field enter, “escalation”
   e) Select High in the priority field
   f) In the Activity Pattern Dates for Target section, enter 10 in the days field.
   g) In the Activity Pattern Dates for Escalation section, enter 30 in the days field.
   h) Click the Update button. If you get a warning about localization, just click the update button again to ignore the warning.
2. **Create a trouble ticket preupdate rule that creates the escalation activity**
   a) In Studio, open the Trouble Ticket pre-update ruleset.
   b) Add a new rule named TTPU1000 – Escalation Activity

```
USES:
uses gw.api.web.admin.ActivityPatternsUtil

CONDITION: (troubleTicket : entity.TroubleTicket):
return !troubleTicket.New and troubleTicket.Escalated and
    !troubleTicket.Activities.hasMatch(\act -> act.ActivityPattern.Code == "escalation")
ACTION (troubleTicket : entity.TroubleTicket, actions : gw.rules.Action) :
  // Create an escalation activity
  var act = new Activity()
  act.ActivityPattern = ActivityPatternsUtil.getActivityPattern("escalation")
```

```
  act.Subject = "Trouble ticket " + troubleTicket.TroubleTicketNumber + " escalated"
  act.Description = "Trouble ticket " + troubleTicket.TroubleTicketNumber +
    " assigned to " + troubleTicket.AssignedUser + "/" + troubleTicket.AssignedGroup
    + " has been escalated"

  // assign the escalation activity to the supervisor of the assigned group
  act.AssignedGroup = troubleTicket.AssignedGroup
  act.AssignedUser = troubleTicket.AssignedGroup.Supervisor

  // associate the escalation activity to the trouble ticket
  troubleTicket.addToActivities(act)
END
```

**3. Restart the server because a new rule was created.**

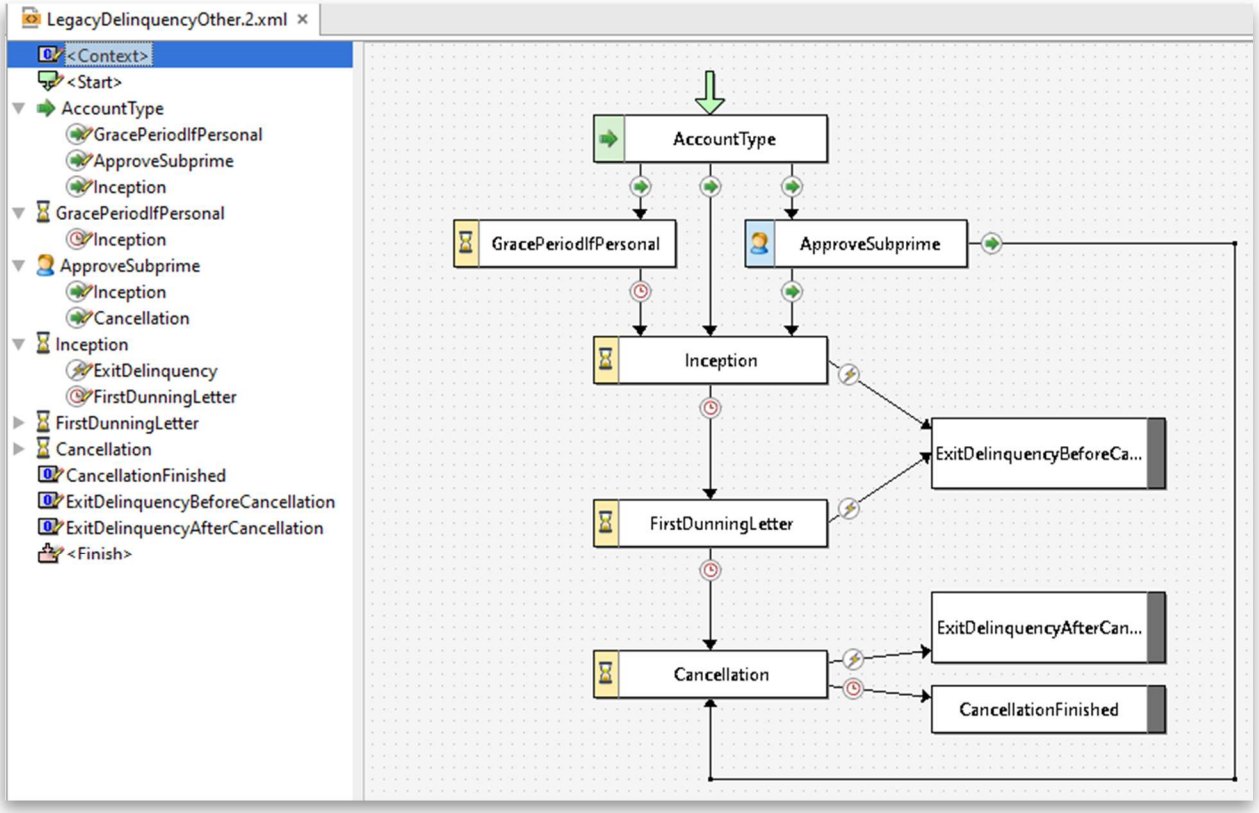
Lesson 3

# Configuring Delinquency Workflow

## 3.1 Modify delinquency workflow

Succeed Insurance Billing Group wants their DP01 delinquency plan to include the option to declare an account delinquent resulting from repeated non-sufficient funds (NSF) payments. Delinquency actions will be based on the account segment (personal, subprime, or other) and will follow a simple workflow consisting of the following events: inception stated, late fee, dunning letter, and cancellation.

This is an overview of what you will add to the workflow to implement the requirement:




### 3.1.1 Requirements

**Spec 1** Delinquency should be invoked upon the second occurrence of a reversal of a direct bill payment within a 30 day period.

**Spec 2** Invoking the delinquency (that is, progressing to the Inception step in the workflow) depends on the account segment type:

a. Subprime accounts require a high-priority activity being auto-assigned and approved before workflow inception. If not approved, move directly to Cancellation.

b. Personal accounts are allowed a grace period of 1 day before workflow inception.

c. Other account types move immediately to workflow inception.

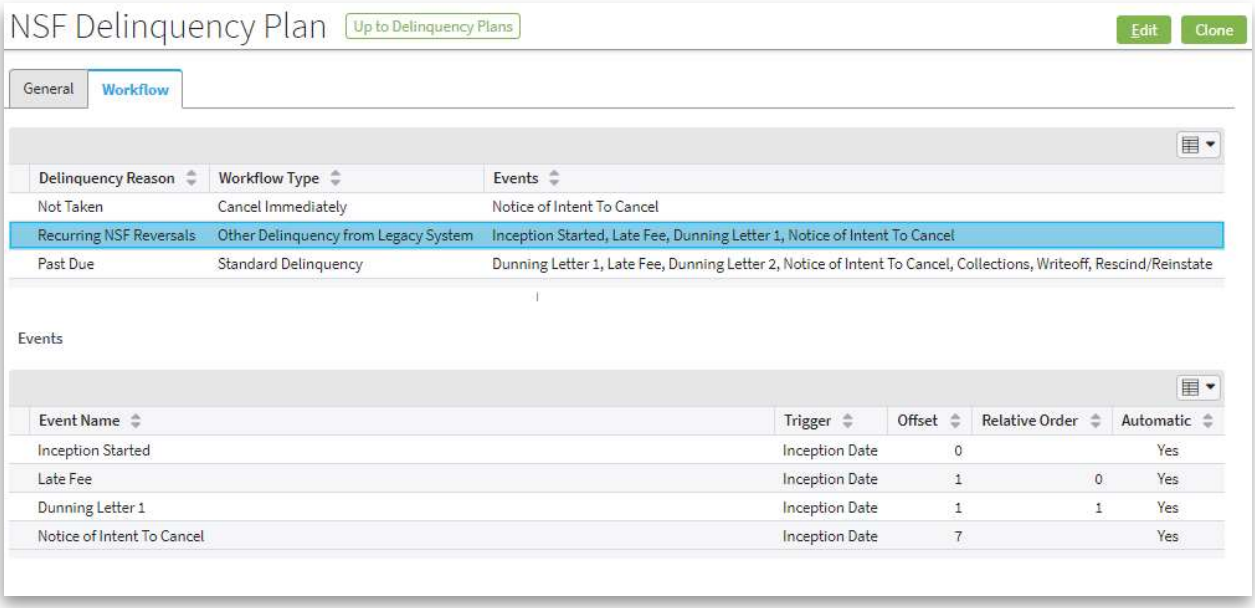
**Spec 3** Grace period length should not be obtained from the delinquency plan since it is used for Past Due delinquencies as well. Use a delta time value of 1 day in the timeout branch.

**Spec 4** When the Inception step is reached, a new event (InceptionStarted) should be flagged as complete.

**Spec 5** When the branch from Inception to First Dunning Letter is executed, a fee should be charged (you can use the Late Fee on the delinquency plan).

**Spec 6** All events should be flagged complete at the appropriate time.

**Spec 7** Create a new delinquency plan called NSF Delinquency Plan based on a clone of DP01. Add workflow types and events as indicated below:




## Hints

Helpful hints that help you to completion.

1. **To assist in launching the delinquency, the number of payment reversals is available on the Account method getNumberOfPejorativePaymentReversals(). Only a reversal due to a returned**

**check is considered a pejorative reversal. Note that the number of pejorative payments reversals does not increase until after the preupdate rule is executed.**

2. **The configuration team suggests implementing this requirement using a BaseMoneyReceivedPreupdate rule.**
3. **The configuration team recommends creating a new version of the LegacyDelinquencyOther workflow as a starting point since it has most of the required steps and actions.**

## 3.1.2 Tasks

1. **Create new delinquency reason.**
2. **Create a new delinquency event name.**
3. **Create a preupdate rule.**
4. **Create a new workflow process of LegacyDelinquencyOther.**
5. **Modify LegacyDelinquencyOther[v2] workflow.**
   a) Create ApproveSubprime activity step.
   b) Modify ApproveSubprime activity.
   c) Add approval condition to Inception go branch
   d) Create go branch from ApproveSubprime step to cancellation step.
   e) Create AccountType auto step.
   f) Modify <Start> block.
   g) Add condition to ApproveSubprime go branch.
   h) Create GracePeriodIfPersonal manual step.
   i) Create go branch from AccountType step to GracePeriodIfPersonal step.
   j) Add condition to GracePeriodIfPersonal go branch.
   k) Create go branch from AccountType to Inception step.
   l) Modify Inception step to create a new flagged event called InceptionStarted.
   m) Modify FirstDunningLetter timeout branch.
6. **Restart server.**
7. **Create new delinquency plan called NSF Delinquency Plan.**

## 3.1.3 Testing procedure

8. **Create an account and associate it with NSF delinquency plan.**
   n) QuickJump: Run Account
   o) Change the delinquency plan
9. **From Actions → New Payment → New Direct Bill Payment, make a direct bill payment for $300 and execute it without distribution.**

10. **From the Account → Payments screen, click on Actions and select Reverse. Select Returned Check as the reason.**
11. **From the Account → Delinquencies screen, confirm the delinquency workflow is not invoked**
12. **Make a direct bill payment for $400 this time, and then reverse that payment.**
13. **From the Account → Delinquencies screen, confirm the delinquency workflow was started and confirm the current event is InceptionStarted.**
14. **Run the Workflow batch process and confirm the Inception Started event is complete (Account tab→Delinquencies).**
15. **Advance the clock to the target date for Dunning Letter 1 and run the Workflow batch process. Has the late fee been charged?**

    p) Open Server Tools (ALT-SHIFT-T)

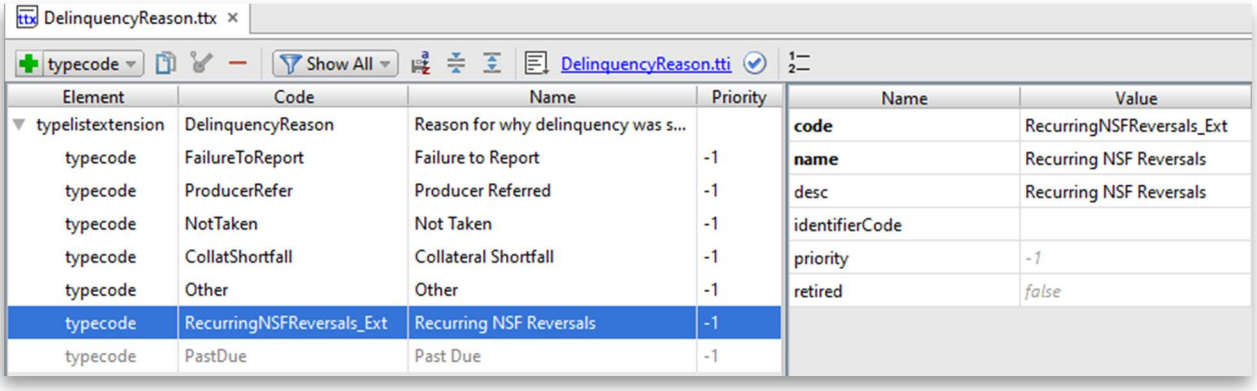
    q) Advance the clock by one day from Internal Tools → Testing System Clock

    r) Run the Workflow process from Server Tools → Batch Process Info

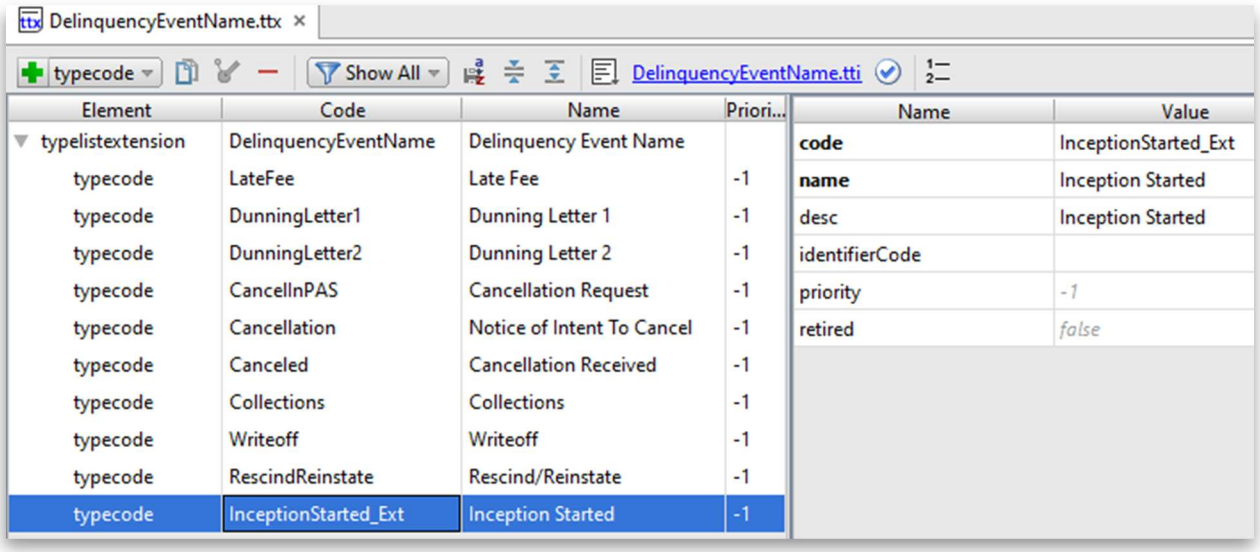
16. **Repeat steps above to test Personal and/or Subprime segments. Note, the Account must be created manually in order to select the desired segment.**

## 3.1.4 Solution

1. **Create new delinquency reason.**




2. **Create a new delinquency event name.**



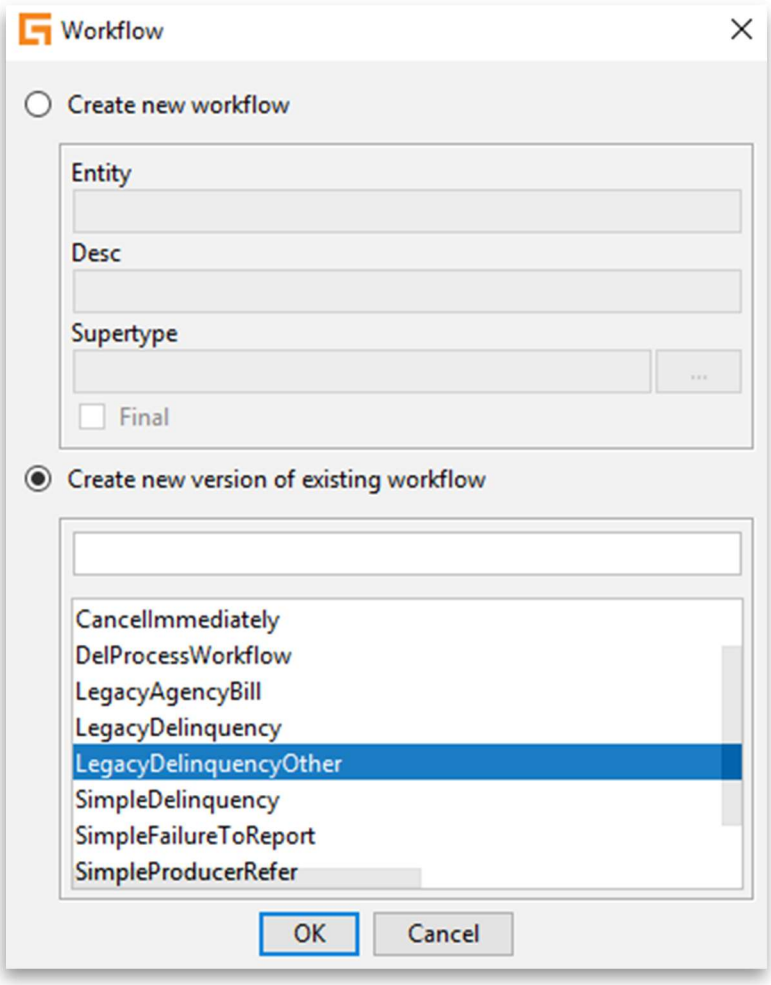

**3. Create a preupdate rule.**

a) In Studio, open the BaseMoneyReceivedPreupdate ruleset.

b) Add a new rule named BMRPU1000 – Second NSF Reversal

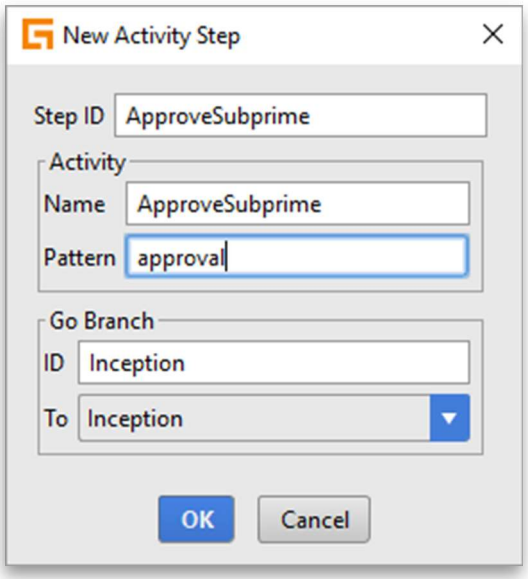
```
USES

CONDITION (baseMoneyReceived : entity.BaseMoneyReceived):
  return
    (baseMoneyReceived.Reversed
    and baseMoneyReceived.Subtype == typekey.BaseMoneyReceived.TC_DIRECTBILLMONEYRCVD
    and (baseMoneyReceived as
    DirectBillMoneyRcvd).Account.getNumberOfPejorativePaymentReversals(30) > 0)
ACTION (baseMoneyReceived : entity.BaseMoneyReceived, actions : gw.rules.Action):
    var account = (baseMoneyReceived as DirectBillMoneyRcvd).Account
    // if no active delinquencies, then start delinquency
    if(!account.hasActiveAccountLevelDelinquencyProcesses()) {
      account.startDelinquency(DelinquencyReason.TC_RECURRINGNSFREVERSALS_EXT)
      // check if active delinquency is recurring NSF reversals delinquency
    } else {
      var dp = account.ActiveDelinquencyProcesses.firstWhere(\p -> p.Reason ==
               DelinquencyReason.TC_RECURRINGNSFREVERSALS_EXT)
      // if not recurring NSF reversals, then start delinquency
      if(dp == null) {
        account.startDelinquency(DelinquencyReason.TC_RECURRINGNSFREVERSALS_EXT)
      }
    }
END
```

**4. Create a new workflow process of LegacyDelinquencyOther.**

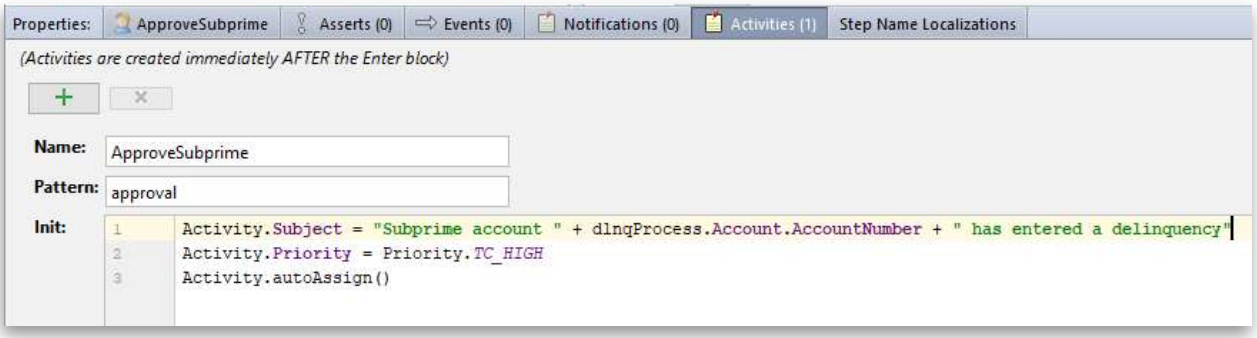



**5. Modify LegacyDelinquencyOther[v2] workflow.**

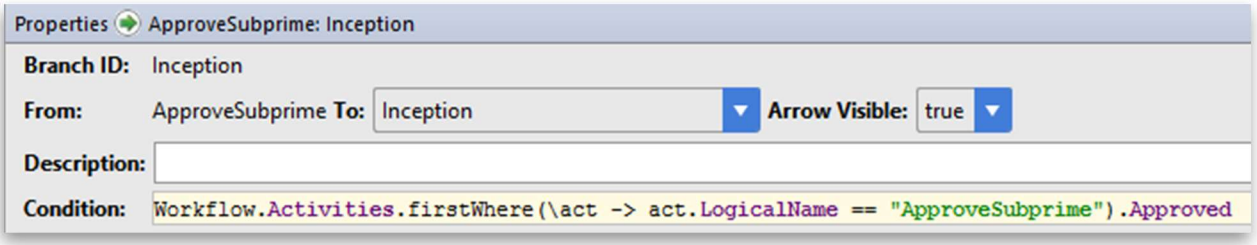
a) Create ApproveSubprime activity step.



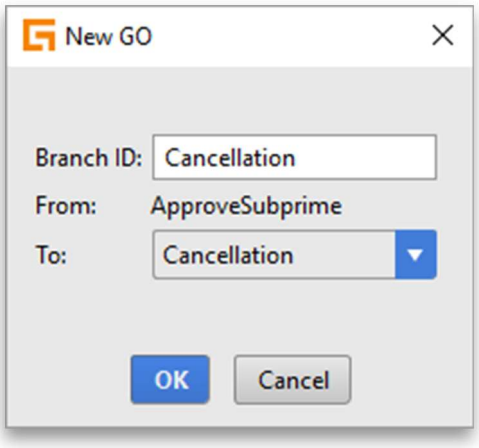

b) Modify ApproveSubprime activity.



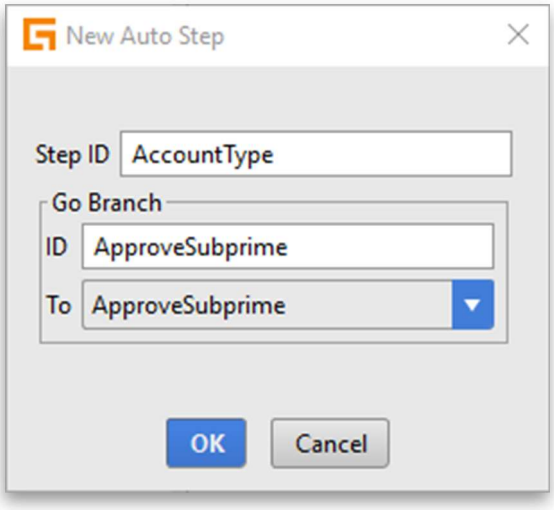

c) Add approval condition to Inception go branch



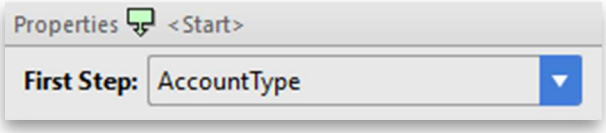

d) Create go branch from ApproveSubprime step to cancellation step.



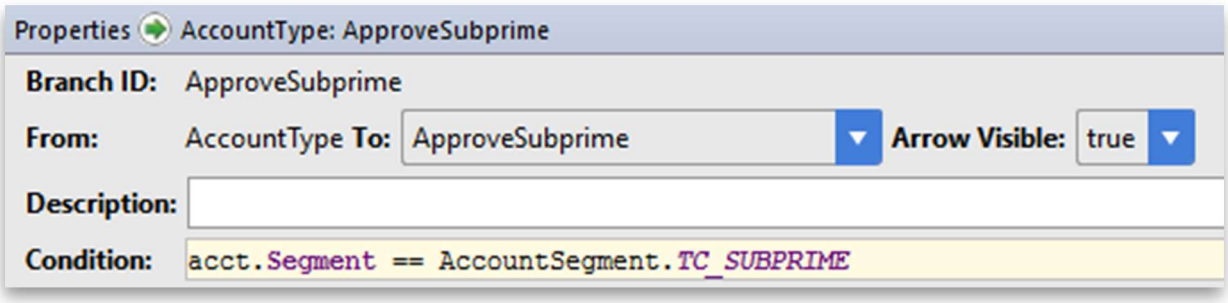

e) Create AccountType auto step.



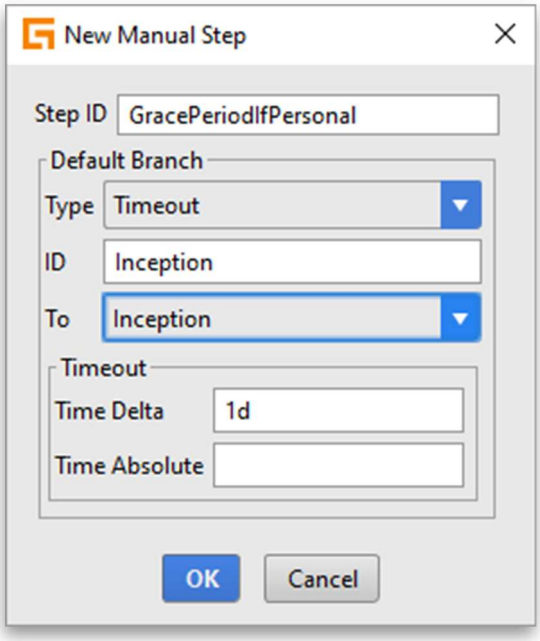

f) Modify <Start> block.



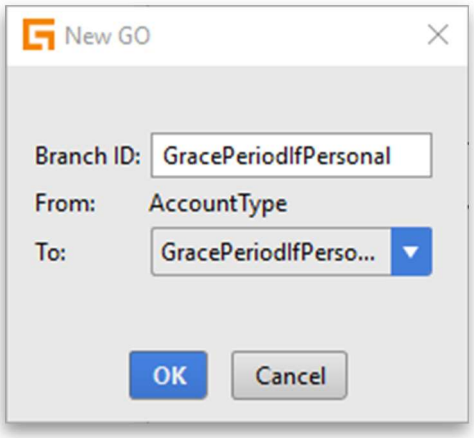

g) Add condition to ApproveSubprime go branch.



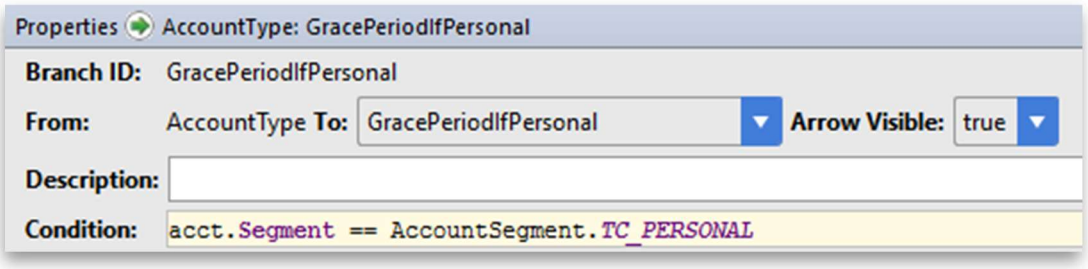

h) Create GracePeriodIfPersonal manual step.



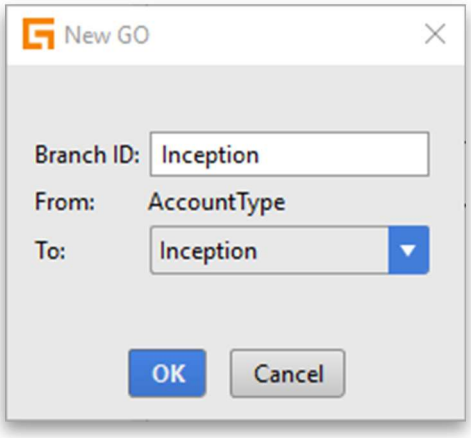

i) Create go branch from AccountType step to GracePeriodIfPersonal step.



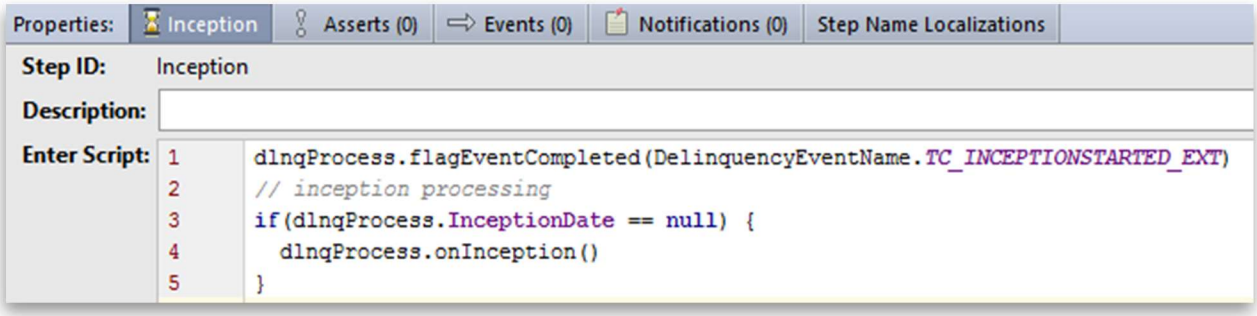
j) Add condition to GracePeriodIfPersonal go branch.



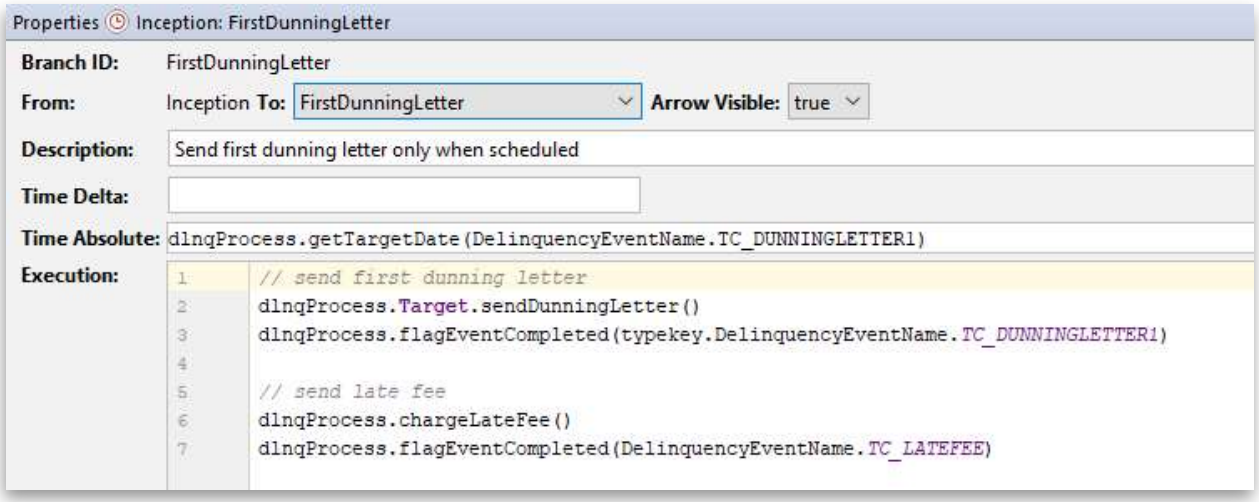

k) Create go branch from AccountType to Inception step.




l) Modify Inception step to create a new flagged event called InceptionStarted.



m) Modify FirstDunningLetter timeout branch.




**6. Restart server.**

**7. Create new delinquency plan called NSF Delinquency Plan.**

a) Clone the DP01 delinquency plan

b) In the new plan, enter "NSF Delinquency Plan" in the name field.

c) Click on the Workflow tab

d) Click the upper add button

e) Change the delinquency reason of the new row to "Recurring NSF Reversals"

f) Change the Workflow field to "Other Delinquency from Legacy System"

g) Add the following events:

| Event Name | Trigger | Offset | Relative Order | Automatic |
|---|---|---|---|---|
| **Inception Started** | Inception Date | 0 | | Yes |
| **Late Fee** | Inception Date | 1 | 0 | Yes |
| **Dunning Letter 1** | Inception Date | 1 | 1 | Yes |
| **Notice of Intent To Cancel** | Inception Date | 7 | | Yes |

h) Click the Update button. If you get a warning about localization, just click the update button again to ignore the warning.

Lesson 4

# Configuring Policy Transactions After Issuance

## 4.1 Configure a new return premium allocation

When BillingCenter receives a credit, Succeed Insurance wants to allocate the credit to the same policy period as the credit charge. Any remaining funds should be distributed first to last.

### 4.1.1 Requirements

**Spec 1** Credits should be allocated to the same policy period first to last.

**Spec 2** Remaining funds should be distributed to other policy periods first to last.

### 4.1.2 Tasks

1. **Add a typecode to the ReturnPremiumAllocateMethod typelist.**
   a) Extend the ReturnPremiumAllocateMethod.tti typelist.
   b) Suggested typecode name: PolicyPeriodFirst_Ext
   c) Do not restart the server until all configuration is done, otherwise, the system will throw an exception.
2. **Create a new class based on ProportionalReturnPremiumAllocationStrategy.gs.**
   a) Suggested package name: si.bc.classes.creditallocation
   b) Suggested class name: PolicyPeriodFirstReturnPremiumAllocationStrategy_Ext
3. **Register the new class in LinkedImplementationLoaderImpl.gs**
4. **Restart the server.**

### 4.1.3 Testing procedure

1. **Create a new Return Premium Plan called RPP CH Lab**
   a) Clone the Default Return Premium Plan
   b) Set the name field to RPP CH Lab
   c) Change the Eligible Items dropdown to Same Payer

d) Add a Policy Change context, select Policy Period First for the method, and move it to top priority

e) Click the Update button to save the new plan

f) Navigate to the Return Premium Plans (Administration → Return Premium Plans) and move the RPP CH Lab plan to top priority

**2. Create an account.**

a) QuickJump: Run Account

**3. Add a policy to the account with these details:**

a) Actions → Add Policy

b) Policy Number: CH-A

c) Payment Plan: PP04

d) Premium: $600

**4. Add a second policy to the new account with these details:**

a) Actions → Add Policy

b) Policy Number: CH-B

c) Payment Plan: PP04

d) Premium: $600

**5. Make a Direct Bill Payment and override the allocation to include any invoice.**

a) Actions → New Payment → New Direct Bill Payment

b) Amount: $360

c) Click Override Distribution

d) Include Only: Up to amount under contract

e) Click Execute

**6. Look at the Charges screen. Note that the payment was applied to both policies.**

**7. Navigate to CH-A policy and create a policy change with a $300 credit**

a) Actions → Change Policy

b) No changes are needed in step 1 of the Policy Change Wizard, so just click Next

c) On step 2, click the add button, select Premium as the type, and enter -300 in the amount field

d) Click Finish to complete the wizard

**8. Navigate to the account and look at the Charges screen. Credit should only be allocated to policy CH-A First to Last.**

a) Make sure CH-B did not receive any credit.

**9. Navigate to CH-A policy and create a policy change with a $200 credit**

a) Actions → Change Policy

b) No changes are needed in step 1 of the Policy Change Wizard, so just click Next

c) On step 2, click the add button, select Premium as the type, and enter -200 in the amount field

d) Click Finish to complete the wizard

10. **Navigate to the account and look at the Charges screen. CH-A should be paid in full and the remaining $80 credit to policy CH-B First to Last.**

### 4.1.4 Solution

1. **Add a typecode to the ReturnPremiumAllocateMethod typelist.**

   a) Extend the ReturnPremiumAllocateMethod.tti typelist.

   b) Suggested typecode name: PolicyPeriodFirst_Ext

   c) Do not restart the server until all configuration is done, otherwise, the system will throw an exception.

2. **Create a new class based on ProportionalReturnPremiumAllocationStrategy.gs.**

   a) Suggested package name: si.bc.classes.creditallocation

   b) Suggested class name: PolicyPeriodFirstReturnPremiumAllocationStrategy_Ext

```
package si.bc.classes.creditallocation

uses gw.api.web.payment.AllocationPool
uses gw.api.web.payment.ReturnPremiumAllocationStrategy
uses gw.payment.FirstToLastAllocationStrategy

/**
 * Credit Allocation - Configure a new Return Premium Allocation lab
 */
class PolicyPeriodFirstReturnPremiumAllocationStrategy_Ext implements
ReturnPremiumAllocationStrategy{

  override function allocate(list: List<BaseDistItem>, allocationPool: AllocationPool) {
    // determine if any positive distribution items exist
    var posDistItems = list.where(\item -> item.InvoiceItem.Amount.IsPositive)
    if(posDistItems.Empty) {
      return
    }
    // determine if any negative distribution items exist
    var negDistItems = list.where(\item -> item.InvoiceItem.Amount.IsNegative)
    if(negDistItems.Empty) {
      return
    }

    // set up variables
    var negPolicyPeriod = negDistItems[0].PolicyPeriod
    var currency = allocationPool.Currency
    var amountAvailable = allocationPool.GrossAmount
    // Get all positive distribution items that have the same policy period as a negative
endorsement
    var preferredPosDistItems = posDistItems.where(\item -> item.PolicyPeriod == negPolicyPeriod)
    // Get remaining positive distribution items
    var remainingPosDistItems = posDistItems.where(\item -> item.PolicyPeriod != negPolicyPeriod)

    // allocate credit to preferred positive distribution items if some exist
    if(preferredPosDistItems.HasElements) {
      // the allocate method will only use the portion needed
      var preferredAmount = preferredPosDistItems.sum(currency, \item -> item.GrossAmountOwed)
      new FirstToLastAllocationStrategy().allocate(preferredPosDistItems,
AllocationPool.withGross(amountAvailable))
      amountAvailable = amountAvailable - preferredAmount

    }
    // allocate credit to remaining distribution items if some exist and if money remains
    if(amountAvailable.IsPositive and remainingPosDistItems.HasElements) {
      new FirstToLastAllocationStrategy().allocate(remainingPosDistItems,
AllocationPool.withGross(amountAvailable))
    }
  }

  override property get TypeKey(): ReturnPremiumAllocateMethod {
    return ReturnPremiumAllocateMethod.TC_POLICYPERIODFIRST_EXT
  }
}
```

### 3. Register the new class in LinkedImplementationLoaderImpl.gs

```
  override function returnPremiumAllocationStrategies() :
Collection<ReturnPremiumAllocationStrategy> {
    return {
        new FirstToLastReturnPremiumAllocationStrategy(),
        new LastToFirstReturnPremiumAllocationStrategy(),
        new ProportionalReturnPremiumAllocationStrategy(),
        new PolicyPeriodFirstReturnPremiumAllocationStrategy_Ext()
    }
  }
```

### 4. Restart the server.

Lesson 5

# Earning and Tracking Commission

## 5.1 Configuring a commission plan

Succeed Insurance has two underwriting companies that sell personal automobile policies. The commission rates vary depending on the underwriting company, the type of charge, and whether a premium is for initial business or a renewal.

### 5.1.1 Requirements

**Spec 1** Create a commission plan called Succeed CommPlan for the carrier with sub plans that implement commission rates for the two underwriting companies.

**Spec 2** The Default subplan will not be used. For this reason, set the commission rates to 0 and do not add any commissionable items.

**Spec 3** Kendall General and Auto Insurers should appear in the list of underwriting companies on the Subplan Availability tab. Make sure they are listed first above existing underwriting companies.

**Spec 4** For Kendall General, the commission is earned when the policy is bound, whereas, for Auto Insurers, the commission is earned pro rata as each payment is received from the insured.

**Spec 5** Both underwriting companies have only primary producers.

**Spec 6** The commission plan should support all producer tiers.

**Spec 7** Suspend for delinquency should be true.

**Spec 8** Only premium is commissionable.

**Spec 9** The following commission schedule needs to be implemented in BillingCenter.

| Subplan Description | Kendall General | Auto Insurers |
|---|---|---|
| New Auto Premium | 20% | 15% |
| Renewal Auto Premium | 17.5% | 10% |

### 5.1.2 Tasks

1. **Update UWCompany typelist.**

2. **Restart the server.**
3. **Create a new commission plan.**
4. **Edit the default sub plan.**
5. **Add a new subplan for Kendall General – New Auto Premium.**
6. **Add a new subplan for Kendall General – Renewal Auto Premium.**
7. **Add a new subplan for Auto Insurers – New Auto Premium.**
8. **Add a new subplan for Auto Insurers – Renewal Auto Premium.**
9. **Save the new commission plan**

### 5.1.3 Testing procedures

**Auto Insurers Scenario**

1. **Create a producer and associate with the new commission plan**
   a) Use QuickJump to create a new producer. The command is Run Producer
   b) On the new producer screen, click the Edit button to put the producer into edit mode
   c) In the producer code section of the screen, click the Add button
   d) Enter AIP as the producer code
   e) In the Commission Plan field, click the search icon to open the search page
   f) Enter Succeed CommPlan into the name field and click search
   g) Click the Select button to select that commission plan
   h) Make note of the producer name for reference later
   i) Click the Update button to save the changes to this producer
2. **Create a new account.**
   a) Use QuickJump to create a new Account. The command is Run Account.
3. **Add a policy to the new account with these details: (Actions → Add Policy)**
   a) Policy number: AI New Auto
   b) Product: Personal Auto
   c) UW Company: Auto Insurers
   d) Payment plan: PP02
   e) Primary: Producer from step 1
   f) Code: AIP
   g) Premium: $1000
4. **Go to the charges screen (Account tab → Charges) and click the placement date of the down payment invoice item for the premium charge.**

a) Has the correct commission amount been reserved for the item? The amount should be $15.00, which is 15% of the item amount of $100.

b) Has any commission been earned for the item? (That is, is any commission payable?) Is this what you expected? No commission should have been earned yet because the commission is earned on each payment received.

**5. Bill the first invoice**

a) Use System Tools to advance the clock to the bill date of the first invoice

b) Run the Invoice batch process

c) Return to the account

d) Has any commission been earned for the item? Is this what you expected? No commission should have been earned yet because the commission is earned on each payment received.

**6. Partially pay the first invoice by $75 and execute. (Actions → New Payment → New Direct Bill Payment)**

**7. Return to the charges screen and click the placement date of the down payment invoice item.**

a) How much commission has been paid for this item? The amount should be $11.25, which is 15% of the payment amount.

b) Is this what you expected? Yes, because the earning criterion is On Payment Rec'd.

**8. Renew the policy with the details below. (Policy tab → Action → Renew Policy)**

a) Policy number: AI New Auto

b) Product: Personal Auto

c) UW Company: Auto Insurers

d) Payment plan: PP02

e) Primary Producer: Producer from step 1

f) Code: AIP

g) Premium: $1200

**9. On the charges screen (Policy → Charges), click on the event date of the down payment item.**

a) Has the correct commission amount been reserved for the item? The amount should be $12.00, which is 10% of the item amount.

b) Has any commission been earned for the item? No commission should have been earned yet because the commission is earned on each payment received.

**Kendall General Scenario**

**1. Create a producer and associate with the new commission plan**

a) Use QuickJump to create a new producer. The command is Run Producer

b) On the new producer screen, click the Edit button to put the producer into edit mode

c) In the producer code section of the screen, click the Add button

d) Enter KGP as the producer code

e) In the Commission Plan field, click the search icon to open the search page

f) Enter Succeed CommPlan into the name field and click search

g) Click the Select button to select that commission plan

h) Make note of the producer name for reference later

i) Click the Update button to save the changes to this producer

**2. Create a new account.**

a) Use QuickJump to create a new Account. The command is Run Account.

**3. Add a policy to the new account with these details: (Actions → Add Policy)**

a) Policy Number: KG New Auto

b) Product: Personal Auto

c) UW Company: Kendall General

d) Payment Plan: PP02

e) Primary Producer: Producer from step 1

f) Code: KGP

g) Premium: $2000

**4. Go to the charges screen (Account tab → Charges) and click the placement date of the down payment invoice item for the premium charge.**

a) How much commission is payable for this item? $40, which is 20% of the item amount of $200.

**5. Click the placement date for the last invoice item.**

a) Has the commission been earned for this item? Is this what you expected? The commission has been earned in full. This is because the earning criterion for Kendall General is On Binding.

**6. Renew the policy with the details below. (Policy tab → Action → Renew Policy)**

a) Policy number: KG New Auto

b) Product: Personal Auto

c) UW Company: Kendall General

d) Payment plan: PP02

e) Primary Producer: Producer from step 1

f) Code: KGP

g) Premium: $2300

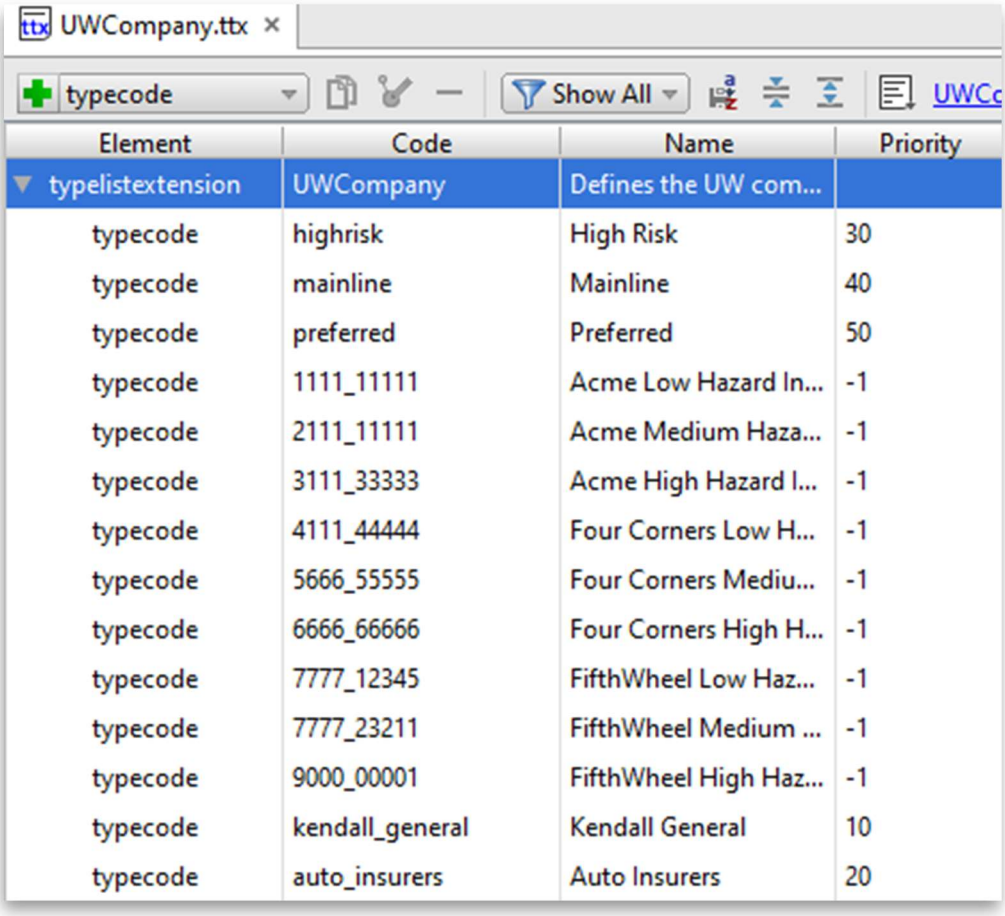
7. **On the charges screen (Policy → Charges), click on the placement date of the down payment item.**

   a) Has the correct commission amount been reserved for the item? The amount should be $40.25, which is 17.5% of the item amount.

   b) Has the commission been earned for this item? Is this what you expected? The commission has been earned in full. This is because the earning criterion for Kendall General is On Binding.

## 5.1.4 Solution

1. **Update UWCompany typelist**




| Element | Code | Name | Priority |
|---|---|---|---|
| typelistextension | UWCompany | Defines the UW com... | |
| typecode | highrisk | High Risk | 30 |
| typecode | mainline | Mainline | 40 |
| typecode | preferred | Preferred | 50 |
| typecode | 1111_11111 | Acme Low Hazard In... | -1 |
| typecode | 2111_11111 | Acme Medium Haza... | -1 |
| typecode | 3111_33333 | Acme High Hazard I... | -1 |
| typecode | 4111_44444 | Four Corners Low H... | -1 |
| typecode | 5666_55555 | Four Corners Mediu... | -1 |
| typecode | 6666_66666 | Four Corners High H... | -1 |
| typecode | 7777_12345 | FifthWheel Low Haz... | -1 |
| typecode | 7777_23211 | FifthWheel Medium ... | -1 |
| typecode | 9000_00001 | FifthWheel High Haz... | -1 |
| typecode | kendall_general | Kendall General | 10 |
| typecode | auto_insurers | Auto Insurers | 20 |

2. **Restart the server.**
3. **Create a new commission plan.**

   a) Navigate to Administration → Business Settings → Commission Plans

   b) Click Actions → New Commission Plan

   c) Enter “Succeed CommPlan” as the name for the new commission plan.

   d) Enter today’s date in the effective date field

e) Select the checkboxes for the Gold, Silver, and Bronze Tiers.

**4. Edit the default sub plan**

a) Enter 0s for the rates

b) In the Earn Commissions dropdown select On Binding. This is a required field.

c) Select Yes for the Suspend for Delinquency field

**5. Add a new subplan for Kendall General – New Auto Premium.**

a) Click the Add button in the sub plans section of the screen

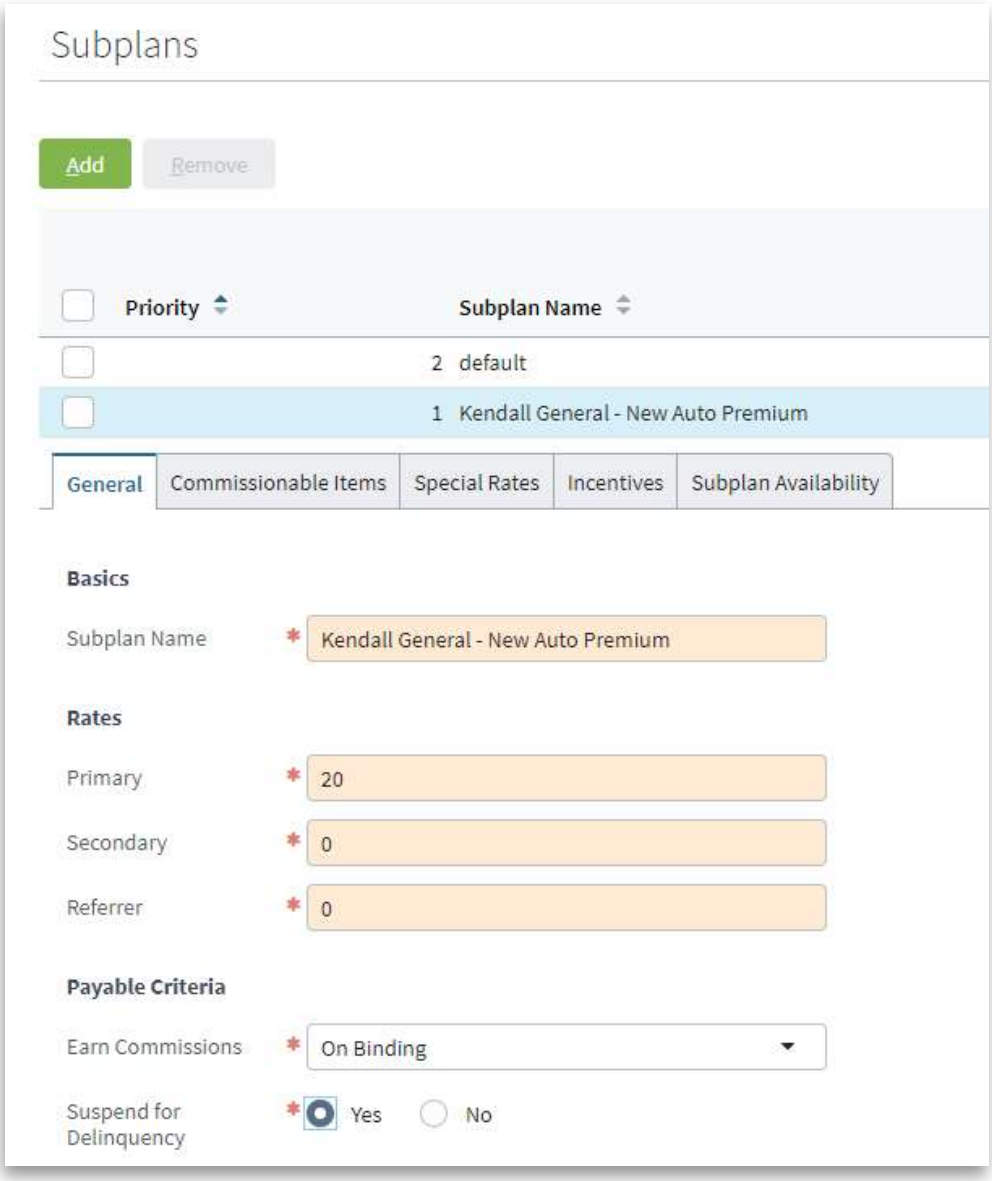
b) Enter "Kendall General – New Auto Premium" as the subplan name

c) Enter 20 in the Primary field

d) Enter 0 in both the Secondary and Referrer fields

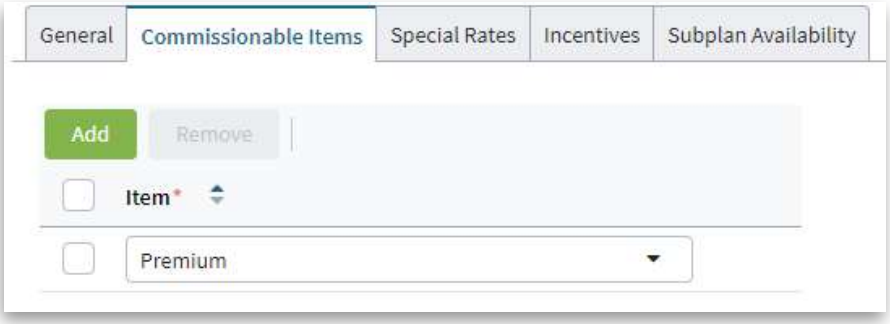
e) In the Earn Commissions dropdown, select On Binding

f) Select Yes for Suspend for Delinquency




g) On the Commissionable Items tab, click the Add button

h) Select Premium from the drop-down list

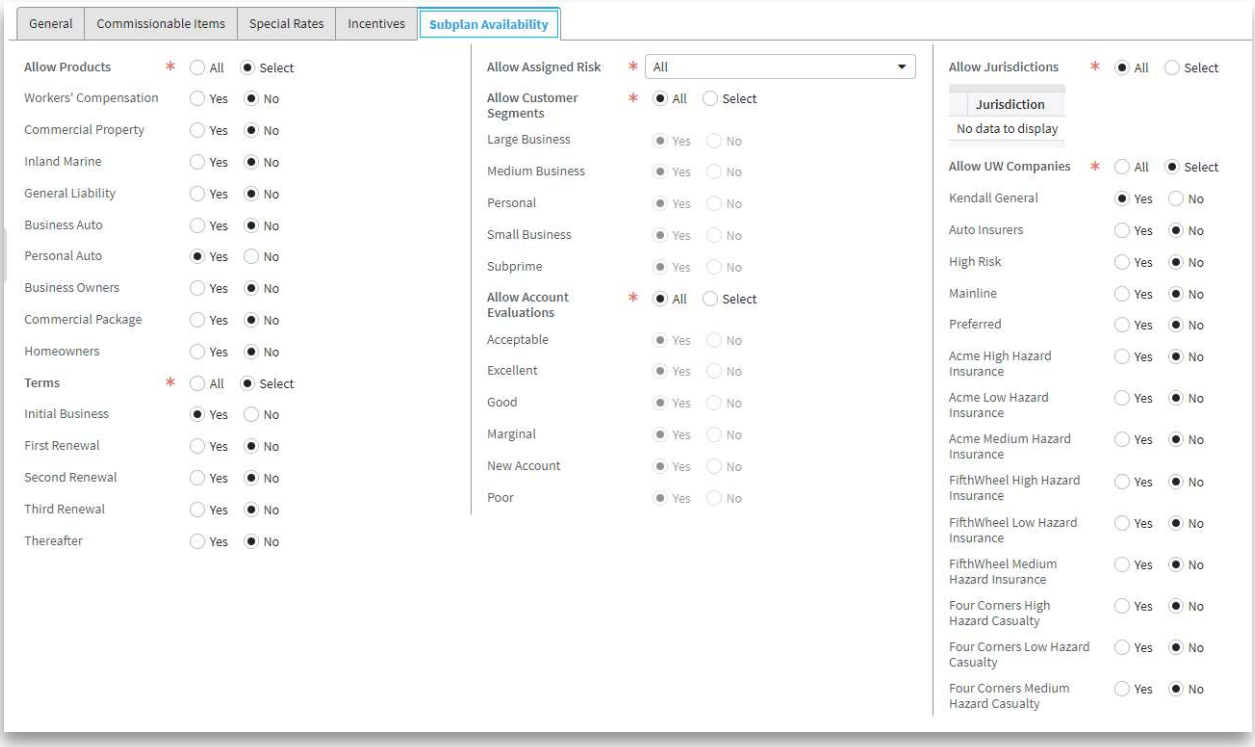



i) Select the subplan availability tab

j) In the Allow Products section, click the Select radio button and then select No for all products EXCEPT for personal auto. Only personal auto should be set to Yes.

k) In the Terms section, click the Select radio button and then select No for all terms EXCEPT for initial business. Only initial business should be set to Yes.

l) In the Allow UW Companies section, click the Select radio button and then select No for all underwriting companies EXCEPT for Kendall General. Only Kendall General should be set to Yes.




**6. Add a new sub plan for Kendall General – Renewal Auto Premium.**

a) Click the Add button in the sub plans section of the screen

b) Enter "Kendall General – Renewal Auto Premium" as the sub plan name

c) Enter 17.5 in the Primary field

d) Enter 0 in both the Secondary and Referrer fields

e) In the Earn Commissions dropdown, select On Binding

f) Select Yes for Suspend for Delinquency

g) On the Commissionable Items tab, click the Add button

h) Select Premium from the drop-down list

i) Select the subplan availability tab

j) In the Allow Products section, click the Select radio button and then select No for all products EXCEPT for personal auto. Only personal auto should be set to Yes.

k) In the Terms section, click the Select radio button and then select No for initial business. First Renewal, Second Renewal, Third Renewal, and Thereafter should all be set to Yes.

l) In the Allow UW Companies section, click the Select radio button and then select No for all underwriting companies EXCEPT for Kendall General. Only Kendall General should be set to Yes.

**7. Add a new subplan for Auto Insurers – New Auto Premium.**

a) Click the Add button in the sub plans section of the screen

b) Enter "Auto Insurers – New Auto Premium" as the subplan name

c) Enter 15 in the Primary field

d) Enter 0 in both the Secondary and Referrer fields

e) In the Earn Commissions dropdown, select On Payment Rec'd

f) Select Yes for Suspend for Delinquency

g) On the Commissionable Items tab, click the Add button

h) Select Premium from the drop-down list

i) Select the subplan availability tab

j) In the Allow Products section, click the Select radio button and then select No for all products EXCEPT for personal auto. Only personal auto should be set to Yes.

k) In the Terms section, click the Select radio button and then select No for all terms EXCEPT for initial business. Only initial business should be set to Yes.

l) In the Allow UW Companies section, click the Select radio button and then select No for all underwriting companies EXCEPT for Auto Insurers. Only Auto Insurers should be set to Yes.

**8. Add a new sub plan for Auto Insurers – Renewal Auto Premium.**

a) Click the Add button in the sub plans section of the screen

b) Enter "Auto Insurers – Renewal Auto Premium" as the sub plan name

c) Enter 10 in the Primary field

d) Enter 0 in both the Secondary and Referrer fields

e) In the Earn Commissions dropdown, select On Payment Rec'd

f) Select Yes for Suspend for Delinquency

g) On the Commissionable Items tab, click the Add button

h) Select Premium from the drop-down list

i) Select the subplan availability tab

j) In the Allow Products section, click the Select radio button and then select No for all products EXCEPT for personal auto. Only personal auto should be set to Yes.

k) In the Terms section, click the Select radio button and then select No for initial business. First Renewal, Second Renewal, Third Renewal, and Thereafter should all be set to Yes.

l) In the Allow UW Companies section, click the Select radio button and then select No for all underwriting companies EXCEPT for Auto Insurers. Only Auto Insurers should be set to Yes.

**9. Save the new commission plan**

a) Click the Update button to save the new commission plan

---

Lesson 6

# Configuring Producer Commission

## 6.1 Implement custom earning criteria

Succeed Insurance sells some policies that have two-year terms. They want to customize how commissions are paid for these policies.

### 6.1.1 Requirements

**Spec 1** Policies are billed every quarter.

**Spec 2** For two year policies only, the primary producer earns commission for all premium charges on binding for invoice items with billing dates that fall within the first year and on the anniversary date of the policy for the remaining commission.

### 6.1.2 Tasks

1. **Configure the getCustomItemCommissionAllocations() method in Commission plugin**
2. **Compile your changes**

### 6.1.3 Testing procedure

1. **Create a commission plan called CustomCriteriaCommissionPlan with these details:**
   a) All tiers are allowed
   b) Only premium is commissionable
   c) Default plan: Primary producer earns 20% using Custom earning criteria
   d) No secondary or referrer producers
   e) Suspend for Delinquency set to Yes
2. **Create a producer (QuickJump: Run Producer) and edit the details and set producer code to CCCP, and the associated commission plan is CustomCriteriaCommissionPlan.**
   a) Note: Record the producer name for reference later.
3. **Create a payment plan by cloning PP09 with these details:**
   a) Name: CCCP 2-year Plan
   b) Max # Installments: 7
4. **Create a new account.**
   a) QuickJump: Run Account

5. **Add a policy to the new account with these details: (Actions → Add Policy)**
   a) Policy Number: CCCP
   b) Expiration Date: 2 years after the Effective Date
   c) Payment Plan: CCCP 2-year Plan
   d) Primary Producer: Producer from step 2
   e) Code: CCCP
   f) Premium: $2400
6. **Go to the charges screen (Account tab → Charges) and click the placement date of the down payment invoice item for the premium charge.**
   a) How much commission is payable for this item? $144, which is 20% of the item amount of $720.
7. **Click on a placement date that is one year past the effective date of the policy.**
   a) Has much commission is payable for this item? None; only a reserve item has been created.
8. **Advance the clock by a one year and run the Commission Payable Calculations batch process**
   a) Open Server Tools (ALT-SHIFT-T)
   b) Advance the clock by one year from Internal Tools → Testing System Clock
   c) Run the Commission Payable Calculations batch process from Server Tools → Batch Process Info
9. **Return the account and click on the last placement date**
   a) How much commission is payable for this item? $48, which is 20% of the item amount of $240.

## 6.1.4 Solution

1. **Configure the getCustomItemCommissionAllocations() method in Commission plugin.**

```
/**
 * Implement Configure Producer Commission demo requirement
 */
override function getCustomItemCommissionAllocations(itemCommissions : Set<ItemCommission>) :
Map<InvoiceItem,MonetaryAmount> {
  // note: by default the "custom" setting merely allocates all unpaid commission
  var today = DateUtil.currentDate()
  var policyEffectiveDate = itemCommissions.first().PolicyCommission.PolicyPeriod.EffectiveDate
  var midTermDate = policyEffectiveDate.addYears(1)
  var allocations = new HashMap<InvoiceItem, MonetaryAmount>()

  // pay year 1 commission
  if (today < midTermDate) {
    foreach (itemCommission in itemCommissions) {
      var invoiceItem = itemCommission.getInvoiceItem()
      var unpaid = itemCommission.CommissionReserve
      if (unpaid.IsNotZero and invoiceItem.InvoiceBilledDate < midTermDate) {
        allocations.put(invoiceItem, unpaid)
      }
    }
  }
  // pay year 2 commission
  else {
    foreach (itemCommission in itemCommissions) {
```

```
      var invoiceItem = itemCommission.getInvoiceItem()
      var unpaid = itemCommission.CommissionReserve
      if (unpaid.IsNotZero) {
        allocations.put(invoiceItem, unpaid)
      }
    }
  }

  return allocations
}
```

2. **Execute Run → Reload Changed Classes**

# 6.2 Implement two-year policy subplan

Succeed Insurance wants to implement the two-year policy commission requirement by adding a subplan to an existing commission plan.

## 6.2.1 Requirements

**Spec 1** A primary producer for policies that are not two-year policies earns 20% commission on the total premium when the first payment is received.

**Spec 2** A primary producer for two-year policies earns 30% commission of the first year's premium immediately, and the remaining commission on the anniversary date of the policy.

**Spec 3** The default value for two-year policies should be true to be consistent with other term values in the subplan screen.

**Spec 4** Make sure to account for two-year policies that include a leap year.

## 6.2.2 Tasks

1. **Add a column to the CommissionSubPlan entity to indicate whether a subplan applies to a two-year policy.**
2. **Restart the server**
3. **Configure the Subplan Availability card of the Commission Plan screen to include a field for the two-year policy indicator.**
4. **Reload the UI**
5. **Configure the selectSubPlan() method in the Commission plugin to select the two-year subplan when the two-year policy condition is met.**
6. **Compile code changes**

## 6.2.3 Testing procedure

Test Two-Year Policies

1. **Clone the CustomCriteriaCommissionPlan created in the previous lab with these details:**
   a) Name: Two-Year Commission Plan
   b) All tiers are allowed

2. **Default plan specific**
   a) Primary producer: 20%
   b) Secondary producer: 0%
   c) Referrer: 0%
   d) Payable Criteria: On First Payment Rec'd
   e) Suspend for delinquency: true
   f) Only premium is commissionable
3. **Two-Year subplan specific**
   a) Name: Two-Year Policy
   b) Primary producer: 30%
   c) Secondary producer: 0%
   d) Referrer: 0%
   e) Payable Criteria: Custom
   f) Suspend for delinquency: true
   g) Only premium is commissionable
4. **Create a producer (QuickJump: Run Producer) and edit the details and set producer code to 2YP, and the associated commission plan is Two-Year Commission Plan.**
   h) Note: Record the producer name for reference later.
5. **Create an account. (QuickJump: Run Account)**
6. **Add a policy to the new account with these details: (Actions → Add Policy)**
   a) Policy Number: 2YP
   b) Expiration Date: 2 years after the Effective Date
   c) Payment Plan: CCCP 2-year Plan
   d) Primary Producer: Producer from step 2
   e) Code: 2YP
   f) Premium: $3000
7. **Go to the charges screen (Account tab → Charges) and click the event date of the down payment invoice item for the premium charge.**
   a) How much commission is payable for this item? $270, which is 30% of the item amount of $900.
8. **Click on an event date that is one year past the effective date of the policy.**
   a) Has much commission is payable for this item? None; only a reserve item has been created.
9. **Advance the clock by a one year and run the Commission Payable batch process**

a) Open Server Tools (ALT-SHIFT-T)

b) Advance the clock by one year from Internal Tools → Testing System Clock

c) Run the Commission Payable batch process from Server Tools → Batch Process Info

**10. Return the account and click on the last event date**

a) How much commission is payable for this item? $90, which is 30% of the item amount of $300.

Test One-Year Policies

**11. Create a producer (QuickJump: Run Producer) and edit the details and set producer code to 1YP, and the associated commission plan is CP01.**

**12. Create a new account.**

a) QuickJump: Run Account

**13. Add a policy to the new account with these details: (Actions → Add Policy)**

a) Policy Number: 1YP

b) Payment Plan: PP02

c) Primary Producer: Producer from step 1

d) Code: 1YP

e) Premium: $800

**14. Go to the charges screen (Account tab → Charges) and click the event date of the down payment invoice item for the premium charge.**

a) How much commission is payable for this item? $8, which is 10% of the item amount of $80.

## 6.2.4 Solution

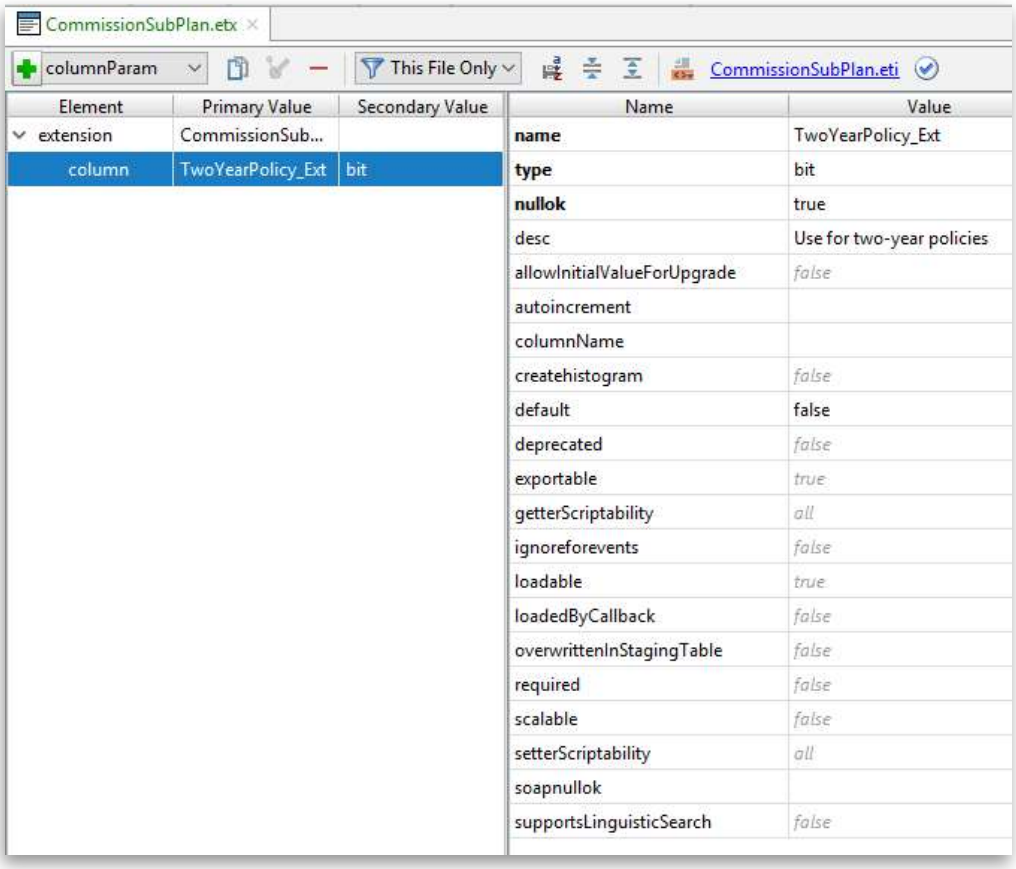
**1. Add a column to the CommissionSubPlan entity to indicate whether a subplan applies to a two-year policy.**

a) Extend the CommissionSubPlan entity.


b) Add a new column.

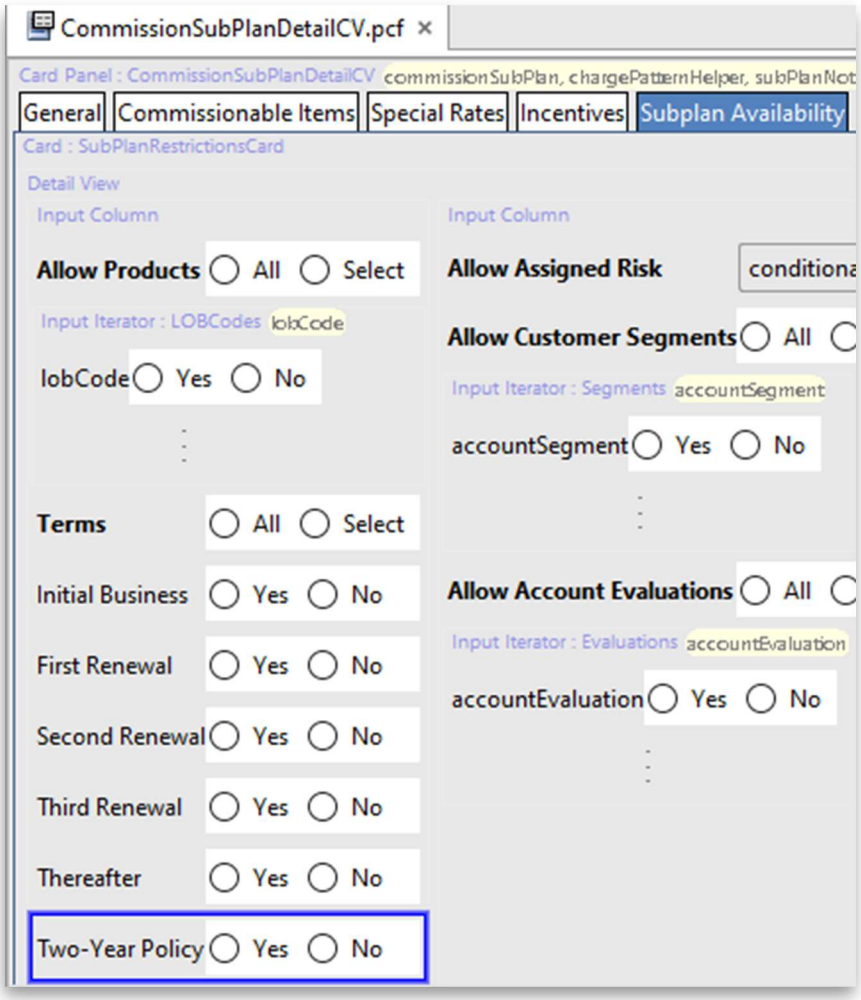


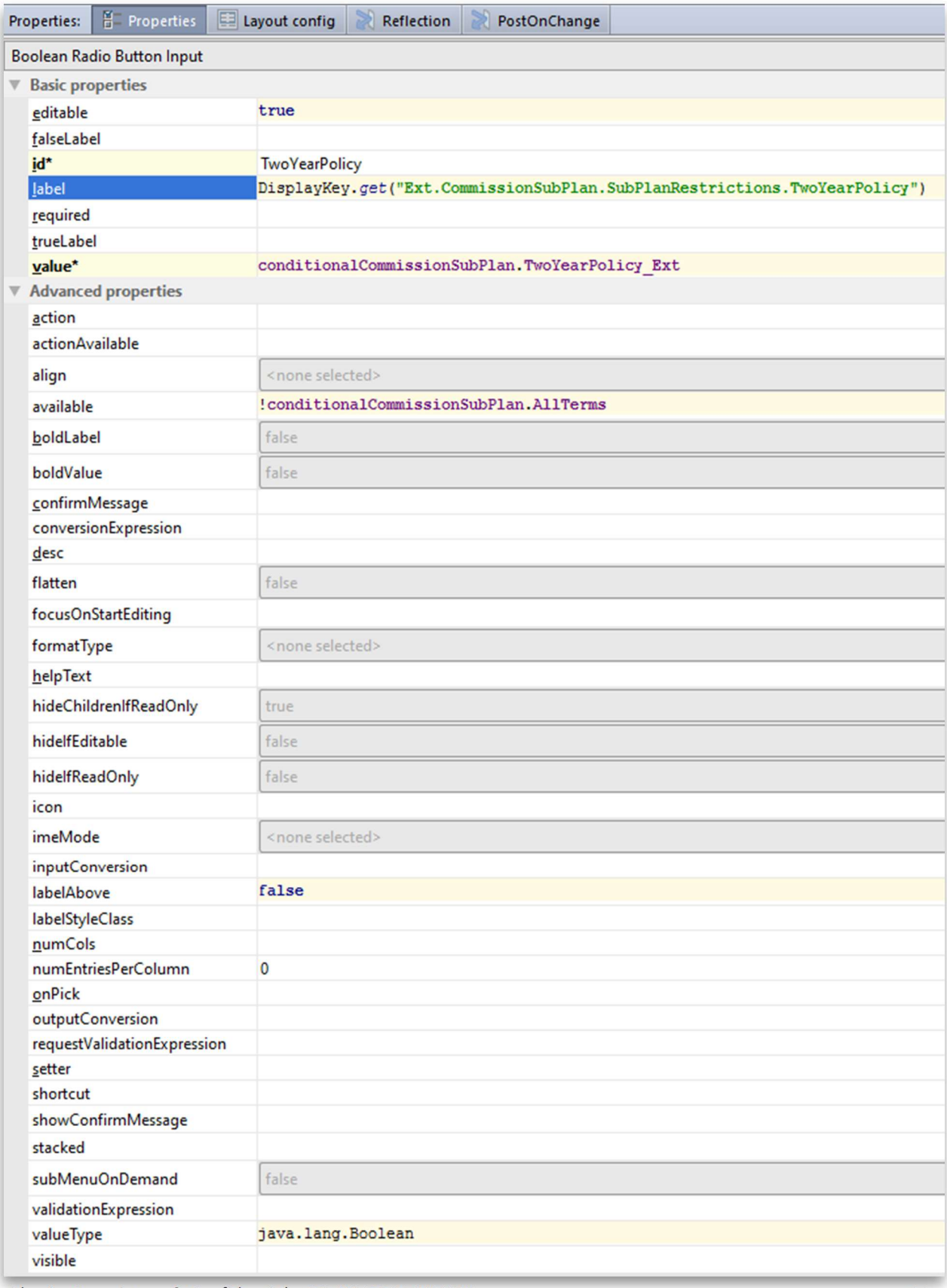
GUIDEWIRE




2. **Restart the server.**
3. **Configure the Subplan Availability card of the Commission Plan screen to include a field for the two-year policy indicator.**

   a) Duplicate the *Thereafter* widget and modify as necessary.


CommissionSubPlanDetailCV.pcf
Card Panel : CommissionSubPlanDetailCV commissionSubPlan, chargePatternHelper, subPlanNot
General
Commissionable Items
Special Rates
Incentives
Subplan Availability
Card : SubPlanRestrictionsCard
Detail View
Input Column
Allow Products All Select
Input Iterator : LOBCodes lobCode
lobCode Yes No
Terms All Select
Initial Business Yes No
First Renewal Yes No
Second Renewal Yes No
Third Renewal Yes No
Thereafter Yes No
Two-Year Policy Yes No
Input Column
Allow Assigned Risk
Allow Customer Segments All
Input Iterator : Segments accountSegment
accountSegment Yes No
Allow Account Evaluations All
Input Iterator : Evaluations accountEvaluation
accountEvaluation Yes No



Properties: | Properties | Layout config | Reflection | PostOnChange

Boolean Radio Button Input

| | |
|---|---|
| **Basic properties** | |
| editable | true |
| falseLabel | |
| **id*** | TwoYearPolicy |
| label | DisplayKey.get("Ext.CommissionSubPlan.SubPlanRestrictions.TwoYearPolicy") |
| required | |
| trueLabel | |
| **value*** | conditionalCommissionSubPlan.TwoYearPolicy_Ext |
| **Advanced properties** | |
| action | |
| actionAvailable | |
| align | <none selected> |
| available | !conditionalCommissionSubPlan.AllTerms |
| boldLabel | false |
| boldValue | false |
| confirmMessage | |
| conversionExpression | |
| desc | |
| flatten | false |
| focusOnStartEditing | |
| formatType | <none selected> |
| helpText | |
| hideChildrenIfReadOnly | true |
| hideIfEditable | false |
| hideIfReadOnly | false |
| icon | |
| imeMode | <none selected> |
| inputConversion | |
| labelAbove | false |
| labelStyleClass | |
| numCols | |
| numEntriesPerColumn | 0 |
| onPick | |
| outputConversion | |
| requestValidationExpression | |
| setter | |
| shortcut | |
| showConfirmMessage | |
| stacked | |
| subMenuOnDemand | false |
| validationExpression | |
| valueType | java.lang.Boolean |
| visible | |

4. **Reload the UI *(ALT+SHIFT+L)*.**
5. **Configure the selectSubPlan() method in the Commission plugin to select the two-year subplan when the two-year policy condition is met.**

```
/*
 * OOTB Method
 /
public override function selectSubPlan(policyPeriod : PolicyPeriod, commissionPlan :
CommissionPlan) : CommissionSubPlan[] {
  return new CommissionSubPlan[0]
}
*/

/**
 * Implement Two-Year Policy Subplan lab
 */
@Param("policyPeriod", "Current policy period")
@Param("commissionPlan", "Commission plan associated with the policy period")
@Returns("An array of applicable subplans")
public override function selectSubPlan(policyPeriod : PolicyPeriod, commissionPlan :
CommissionPlan) : CommissionSubPlan[] {
  // setup date logic
  var daysTwoYears = 730
  var daysTwoYearsWithLeapYear = 731
  var daysPolicyPeriodTerm =
policyPeriod.PolicyPerExpirDate.daysBetween(policyPeriod.PolicyPerEffDate)
  // determine applicable subplans
  if(daysPolicyPeriodTerm == daysTwoYears or daysPolicyPeriodTerm == daysTwoYearsWithLeapYear) {
    return commissionPlan.SubPlans.where(\sp -> sp.TwoYearPolicy_Ext)
  }
  else {
    return commissionPlan.SubPlans.where(\sp -> !sp.TwoYearPolicy_Ext)
  }
}
```

6. **Compile code changes (*Run → Reload Changed Classes).***

## 6.3 Demo code: Configuring commission allocation

The following code is from the instructor demo:

```
/**
* Implement custom earning criterion, where a policy earns all commission 2
* months after policy effective date.
*/
@Param("ItemCommission", "Set of invoice items")
@Returns("A map of MonetaryAmount objects of allocated commission payments")
override function getCustomItemCommissionAllocations(itemCommissions : Set<ItemCommission>) :
Map<InvoiceItem,MonetaryAmount> {

  var today = DateUtil.currentDate()
  var policyEffectiveDate = itemCommissions.first().PolicyCommission.PolicyPeriod.EffectiveDate
  var twoMonthsPastEffectiveDate = policyEffectiveDate.addMonths(2)
  var allocations = new HashMap<InvoiceItem, MonetaryAmount>()

  // Do not allocate any commission before policy is two months past effective date
  if (twoMonthsPastEffectiveDate > today)
    return allocations

  // Pay all commission after two months
  foreach(itemCommission in itemCommissions) {
    var invoiceItem = itemCommission.getInvoiceItem()
```

```
    var unpaid = itemCommission.CommissionReserve
    if (!unpaid.IsZero) {
      allocations.put(invoiceItem, unpaid)
    }
  }
  return allocations
}
```

Lesson 7

# Overriding Policy Commission

## 7.1 Overriding policy commission

In this lab, you will perform the same steps that you saw during the instructor demo.

### 7.1.1 Tasks

1. **Using the policy (1YP) that you used in the last lab, override the commission rate to 5%**
2. **View the code that executes the commission change**

### 7.1.2 Solution

1. **Using the policy (1YP) that you used in the last lab, override the commission rate to 5%**
   a) Search for the policy 1YP from the previous lab
   b) Navigate to the commission page
   c) Click the Override Plan button
   d) Select Percentage from the drop-down list
   e) Enter 5 in the field
   f) Click the Update button
2. **View the code that executes the commission change**
   a) Click the Override Plan button
   b) ALT+SHIFT+E to open the OverrideCommissionPlanPopup.pcf.
   c) View the properties of the PCF. The beforeCommit property calls applyOverrideSettings() method.
   d) CTRL+CLICK on applyOverrideSettings() to navigate to the method.

Lesson 8

# Configuring Policy Transfer

## 8.1 Configure transfer rate for new producer

When a policy is transferred from one producer to another and it is a retroactive transfer, the commission rate on the policy should be automatically adjusted to the new producer's rate.

### 8.1.1 Requirements

**Spec 1** Configure the Transfer Policy wizard so that immediately after a retroactive transfer, the commission rate on the policy and role being transferred is that of the destination producer's commission rate for that role.

### 8.1.2 Configuration notes

Before performing the tasks, please review the following configuration notes:

1. **There are two wizards that perform policy transfer. The wizard that will be configured for this lab is the Transfer Policy wizard (Policy Tab → Commissions → Transfer Policy).**
2. **The beforeCommit property executes policyTransfer.executeTransfer() method during the execution the Transfer Policy Wizard. Create a new pcf function called customCommissionRateTransfer() that adjusts the commission rate for retroactive transfers. Call the new pcf customCommissionRateTransfer() method from the beforeCommit property as well.**
3. **If the policy has no source producer for the role that you are transferring, there is no need to override the commission rate. In this case, the commission rate will be determined by the target producer's applicable subplan.**

### 8.1.3 Tasks

1. **Create a new customCommissionRateTransfer () method in TransferPolicyWizard.pcf file and reference from beforeCommit property.**
2. **Compile code changes.**

### 8.1.4 Testing procedure

1. **Create two new producers (QuickJump: Run Producer) and edit the details:**

**Producer 1**

a) Producer Code: SP

b) Commission Plan: CP03

c) Plan Rate: 10%

**Producer 2**

a) Producer Code: TP

b) CommissionPlan: CP04

c) Plan Rate: 12%

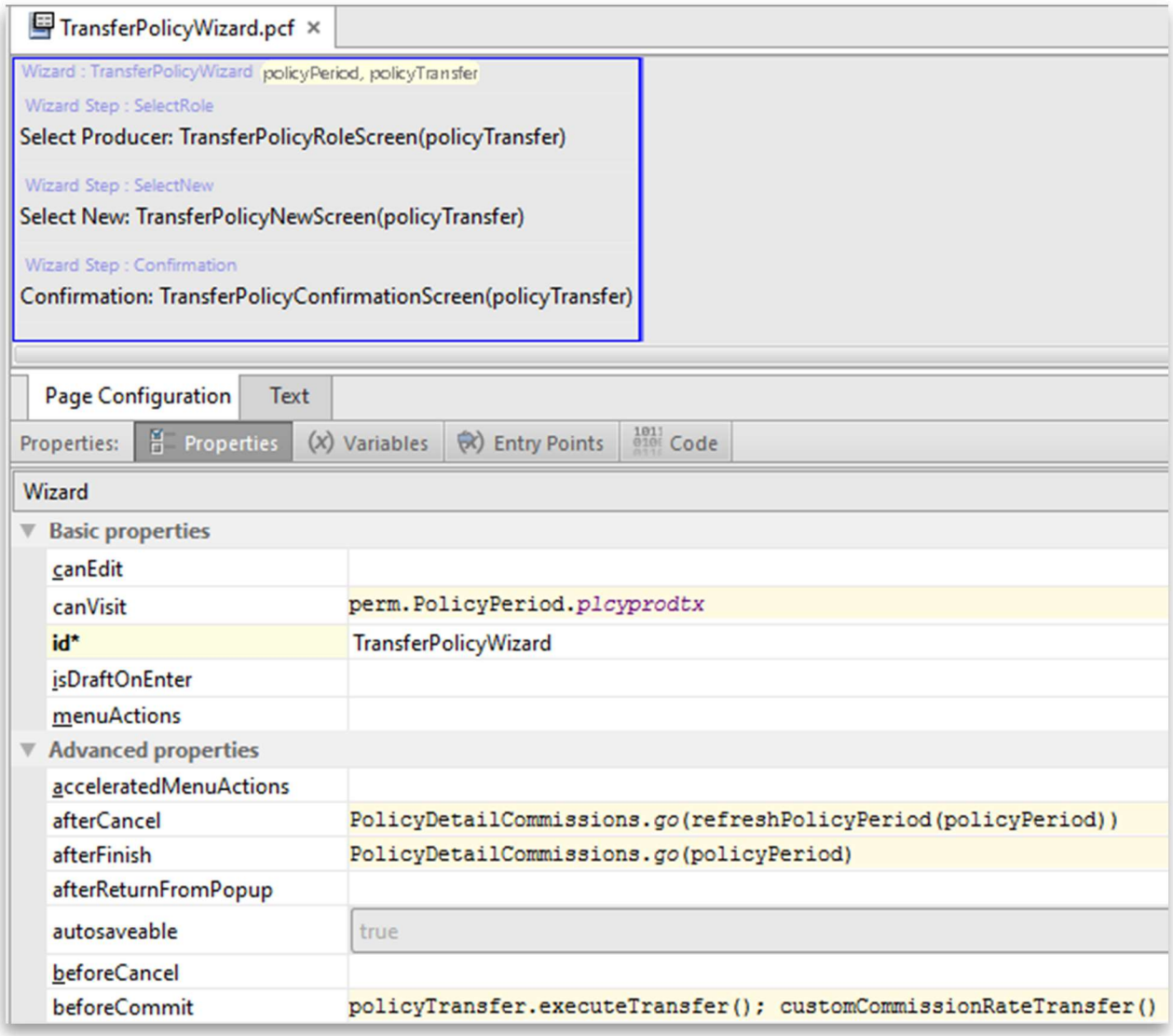
Note: Record the producer names for reference later.

2. **Create a new account.**
   a) QuickJump: Run Account
3. **Add a policy to the new account with these details: (Actions → Add Policy)**
   a) Policy Number: Xfer
   b) Payment Plan: PP04
   c) Primary Producer: Producer 1 from step 1
   d) Code: SP
   e) Premium: $1200
4. **Go to the charges screen (Account tab → Charges) and click the event date of the down payment invoice item for the premium charge.**
   a) How much commission is reserved for this item? $36, which is 10% of the item amount of $360.
5. **Navigate to Xfer policy.**
6. **Execute a policy retroactive transfer from producer 1 to producer 2. (Policy Tab → Commissions → Transfer Policy)**
   a) Role: Primary
   b) Producer: Search for Producer 2
   c) Code: TP
   d) Commission Options: Commissions retroactive to effective date transfer to new producer
7. **On the Commissions screen, verify the producer 2 is the new producer.**
8. **Filter the commission transactions by selecting producer 2.**
   a) How much commission is reserved? $144, which is 12% of the premium amount of $1200.

## 8.1.5 Solution

**1. Create a new customCommissionRateTransfer () pcf method in TransferPolicyWizard.pcf file and reference from beforeCommit property.**

```
/**
 * Implement Configure Transfer Rate for New Producer lab
 */
function customCommissionRateTransfer() : void {
  if (policyTransfer.SourceProducerCode.Producer != null and
      policyTransfer.CommissionTransferOption == CommissionTransferOption.TC_RETROACTIVE) {
    // modify commission rate to that of target producer
    var roleToTransfer = policyTransfer.RoleToTransfer    // this could be Primary, Secondary, or
Referrer
    var newProducerRate = policyTransfer.DestinationProducerCode.CommissionPlan
        .getApplicableSubPlan(policyPeriod)
        .getBaseRate(roleToTransfer)
    for (charge in policyPeriod.Charges) {
      charge.overrideCommissionRate(roleToTransfer, newProducerRate)
    }
  }
}
```




**2. Compile code changes (Run → Reload Changed Classes).**

Lesson 9

# Configuring Transaction Approval

## 9.1 Configure transaction approval

Succeed Insurance want to implement aggregate approval limits for account disbursements. For example, suppose there is a user who has the authority to create disbursements of up to $100. The user wants to create a disbursement for $120 without waiting for approval. To circumvent their authority limit, the user creates two $60 disbursements for an account. Both disbursements are below the user's authority limit, so both are automatically approved.

Succeed wants to prevent the user from doing this by enforcing a user's aggregate approval limit for disbursements. If a user has this limit, then the first $60 disbursement would be approved automatically. But the second disbursement would exceed the aggregate approval limit and therefore require approval.

### 9.1.1 Requirements

**Spec 1** Implement aggregate approval limits that limit account disbursements for any given account.

**Spec 2** Do not enforce aggregate approval limits for automatic disbursements.

**Spec 3** Enforce aggregate approval limits per user

### 9.1.2 Tasks

1. **Find all AuthorityEvent instances linked to the account associated with the current transaction.**
2. **Total the amounts of the transaction objects linked to the relevant AuthorityEvent instances.**
3. **Retrieve the user's aggregate authority limit.**
4. **Check to see if the total of the previous transactions plus the new transaction is less than or equal to the aggregate authority limit.**

### 9.1.3 Testing procedure

1. **Add a test policy**
   a) Create a test account by running the Run Account quick jump command
   a) Add a new policy with $240 premium. Fill in any required fields
2. **Fully pay the policy**
   a) Click Actions → New Payment → New Direct Bill Payment

b) Enter 240 in the amount field

c) Click the Execute Without Distribution button

**3. Create a policy change to reduce the premium for the policy**

a) Navigate to the policy

b) Click Actions → Change Policy

c) On step 1 of the wizard, no additional changes are needed so click Next.

d) In step 2, click the add button and enter -120 in the amount field and then click finish.

**4. Verify that Aaron Applegate has an authority limit of $100 for disbursements**

a) Navigate to Administration → Users & Security → Users

b) Click on Aaron Applegate

c) On the Authority Limits tab, confirm that his Approve Disbursement authority limit is set to $100.

**5. Log in as a limited user and access test account**

a) Log in as Aaron Applegate (aapplegate)

b) Use the account search to locate the test account

**6. Create the first $60 disbursement**

a) Click Actions → New Transaction → Disbursement to start the Account Disbursement Wizard.

b) Enter 60 in the amount field

c) Select the current billing system date as the payment date

d) Select Return Premium PolicyChange as the reason

e) Click next to move to step 2 of the wizard and then click finish

f) Click on Disbursements in the side panel

g) Notice that the disbursement is approved

**7. Create the second $60 disbursement**

a) Click Actions → New Transaction → Disbursement to start the Account Disbursement Wizard.

b) Enter 60 in the amount field.

c) Select the current billing system date as the payment date.

d) Select Return Premium PolicyChange as the reason.

e) Click next to move to step 2 of the wizard.

f) Notice that an alert message is displayed at the top of the screen. This message tells the user that disbursement will require approval because this user does not have sufficient authority.

g) Click finish.

h) Click on Disbursements in the side panel.

i) Notice that the second disbursement is awaiting approval.

## 9.1.4 Solution

**1. Comment out the existing code**

a) Open the ActivityApprovablePlugin.gs in Studio

b) Comment out the existing version of the isCanApproveDisbursement method

**2. Add code to the method as shown below**

```
  override function isCanApproveDisbursement(disbursement: Disbursement,
isCurrentUserCanApproveAction: Boolean): Boolean {
    if (disbursement typeis AccountDisbursement) {
      var totalAccountDisbursementAmountForUser = disbursement.Amount
      var disbUtil = new DisbursementUtil()
      foreach (disb in disbUtil.getAccountAndCollateralDisbursements(disbursement.Account)) {
        if(disb typeis AccountDisbursement) {
          if(disb.AuthorityEvent.CreateUser == User.util.CurrentUser) {
            totalAccountDisbursementAmountForUser += disb.Amount
          }
        }
      }
      var max = User.util.CurrentUser.getAuthorityProfile().
          getMaximumAuthorizedAmount(AuthorityLimitType.TC_APPROVEDISBURSEMENT,
disbursement.Account.getCurrency())
      return totalAccountDisbursementAmountForUser <= max
    }
    return isCurrentUserCanApproveAction
  }
```

**3. Deploy your changes by running Run → Reload Changed Classes.**

Lesson 10

# Configuring Location Groups and Pages

## 10.1Change account tab, policy tab, and info bar behavior

The billing clerks would like to have the Account tab and Policy tab behave like other Guidewire applications. They would also like to have the account segment (such as Medium Business, Subprime, Personal, and so on) displayed next to the account number whenever they are looking at an account.

### 10.1.1 Requirements

**Spec 1** When you click the Account tab, it opens the last account you were working with. If you were not working with an account, clicking the tab opens a page that lets you choose whether to create a new account from the Actions menu or search for an existing account.

**Spec 2** When you click the Policy tab, it opens the last policy you were working with. If you were not working with a policy, clicking the tab opens a page that lets you search for an existing policy.

**Spec 3** On the Account tab, add the account segment information to the info bar. Display the segment information between the account number and date.

### 10.1.2 Tasks

1. **Open the pcf file that controls navigation when the Account tab is clicked.**
2. **Edit the variable that is used in the first Forward Condition element condition expression.**
3. **Repeat tasks for the Policy tab.**
4. **Add a new widget to the info bar associated with the account location group to display the account segment information.**
5. **Deploy code changes.**
6. **Reload UI prior to testing.**

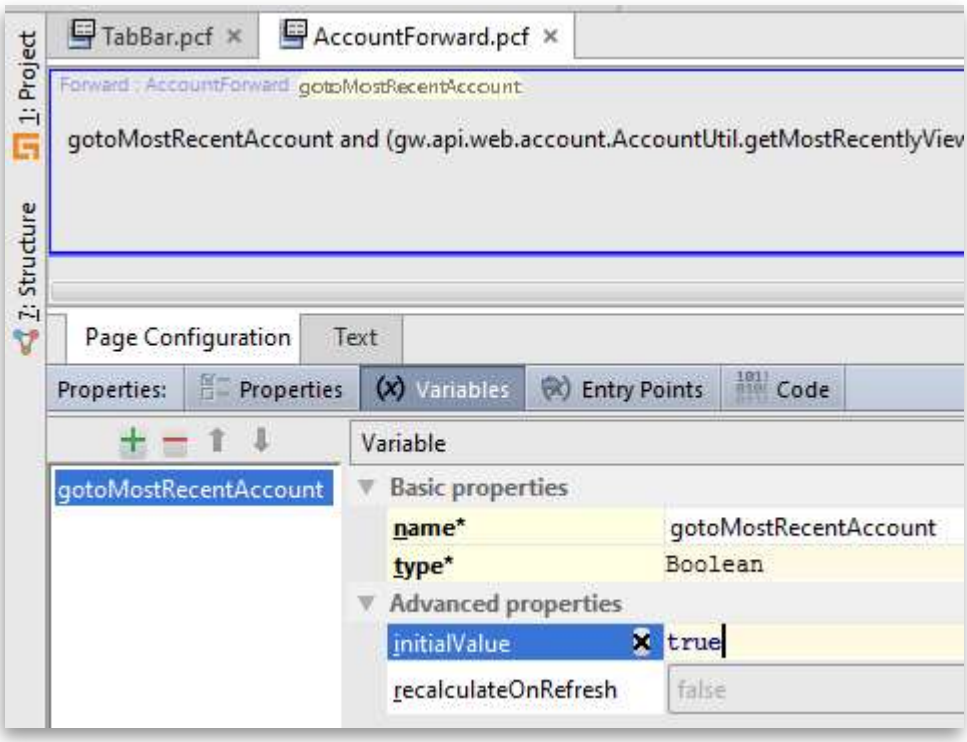
### 10.1.3 Testing procedure

1. **Click on the Account tab and verify that the most recent account was opened.**
2. **Verify that the account segment information is displayed in the info bar.**
3. **Click on the Policy tab and verify that the most recent policy was opened.**

### 10.1.4 Solution

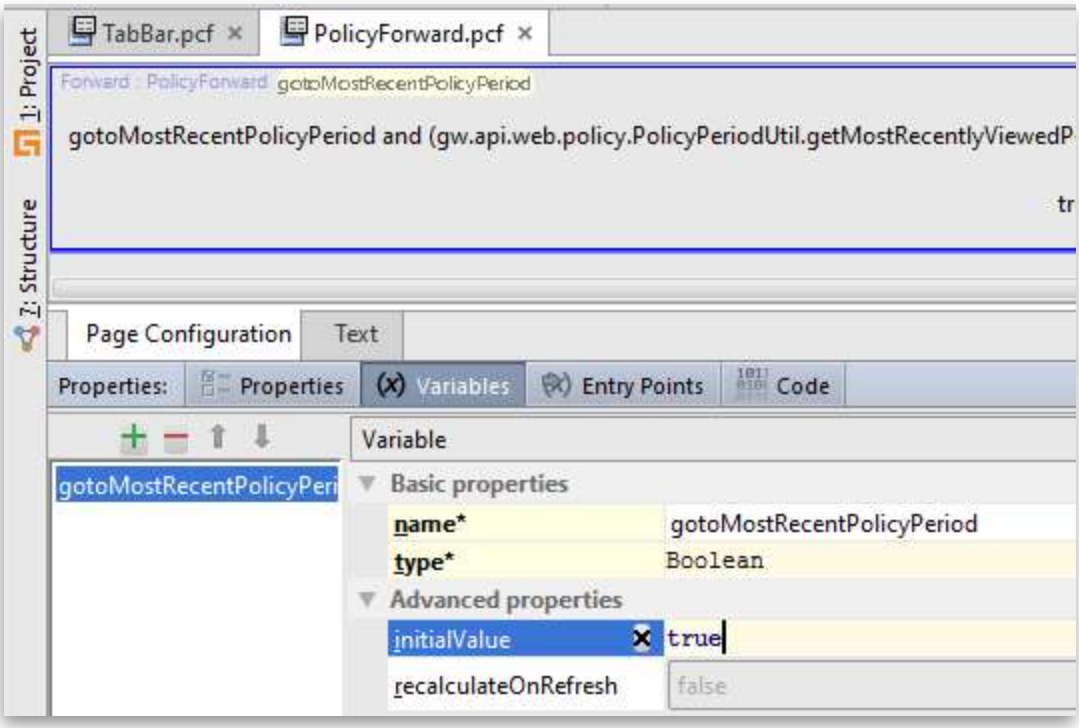
1. **Open the AccountForward.pcf file.**

**2. Edit the gotoMostRecentAccount variable and set the initialValue property to true.**

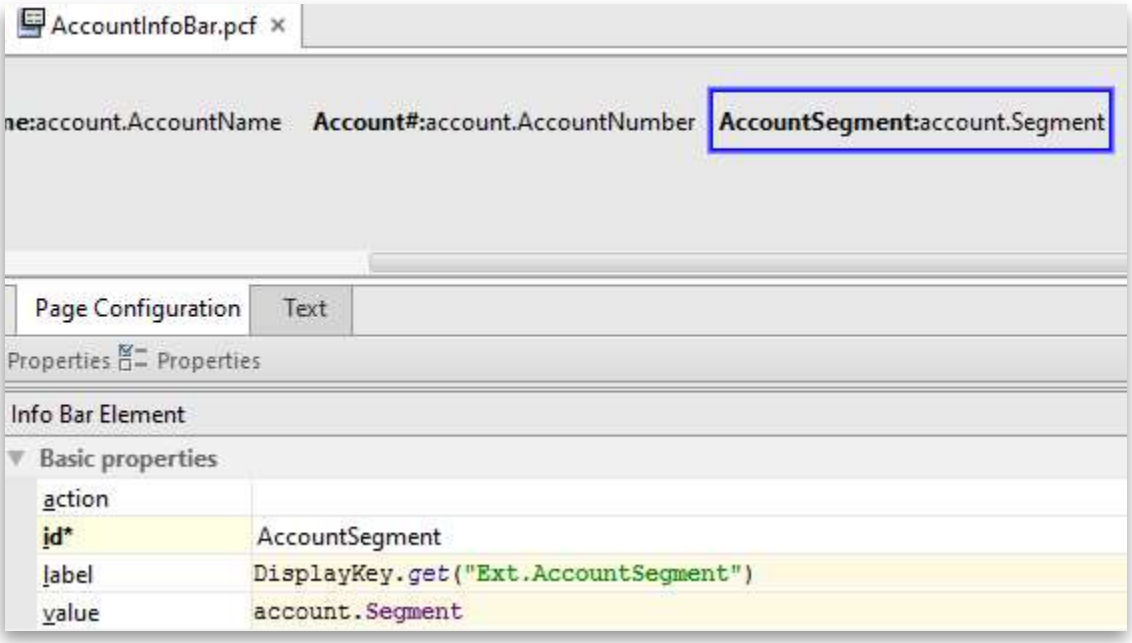



**3. Open the PolicyForward.pcf file.**

**4. Edit the goToMostRecentPolicyPeriod variable and set the initialValue property to true.**




5. **Open the AccountInfoBar.pcf file.**
6. **Duplicate the AccountNumber widget and edit the properties.**




7. **Deploy your changes by running Run → Reload Changed Classes.**
8. **Reload UI by running ALT+SHIFT+L prior to testing.**

# 10.2 Configure a new billing location group

Users want to be able to easily access all billing information (Charges, Invoices, Invoice Streams) for an account.

## 10.2.1 Requirements

**Spec 1** Create a new location group on the Account tab called Billing to contain all billing-related screens.

**Spec 2** Display Charges, Invoices, and Invoice Streams in the new Billing location group.

**Spec 3** Display the Billing link after Funds Tracking and before Payments.

## 10.2.2 Tasks

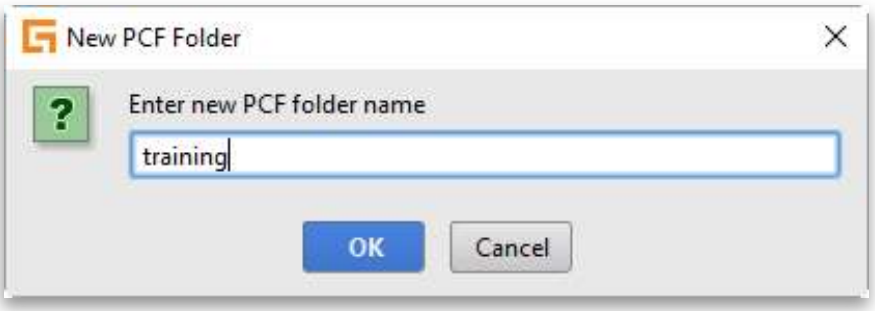
1. **Create a new pcf folder called training.**
2. **Create a new location group pcf file called BillingGroup.**
3. **Set the essential properties, entry point, and variables for BillingGroup pcf file.**
4. **Move the existing location ref for Charges, Invoices, and Invoice Streams to BillingGroup pcf file.**
5. **Add the BillingGroup location group to the account location group.**
6. **Deploy your changes by running Run → Reload Changed Classes.**
7. **Reload UI by running ALT+SHIFT+L prior to testing.**

## 10.2.3 Testing procedure

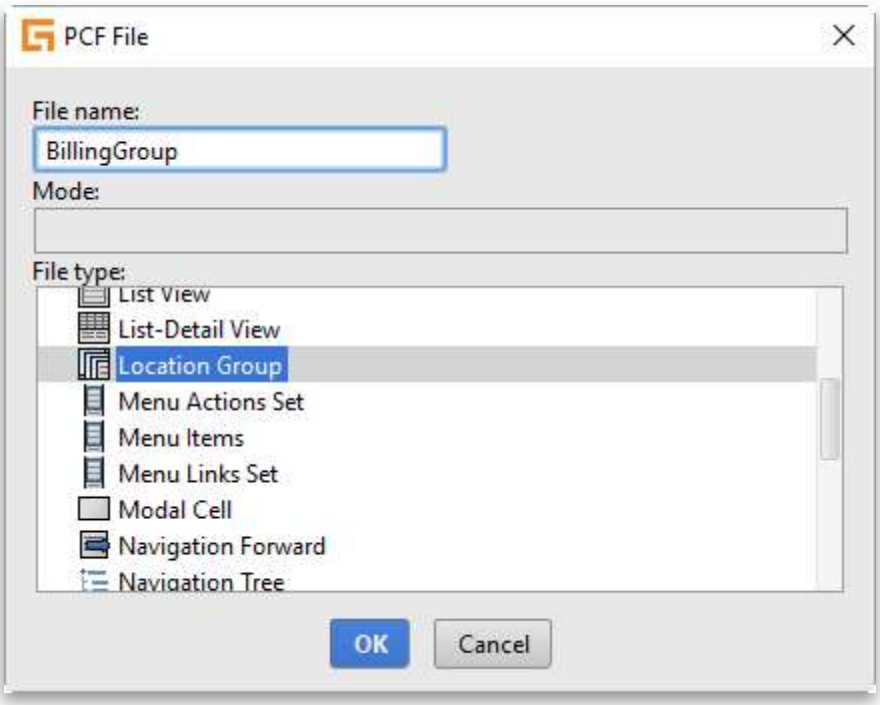
Verify that you can access Charges, Invoices, and Invoice Stream screens from the Billing location group on the account sidebar.

## 10.2.4 Solution

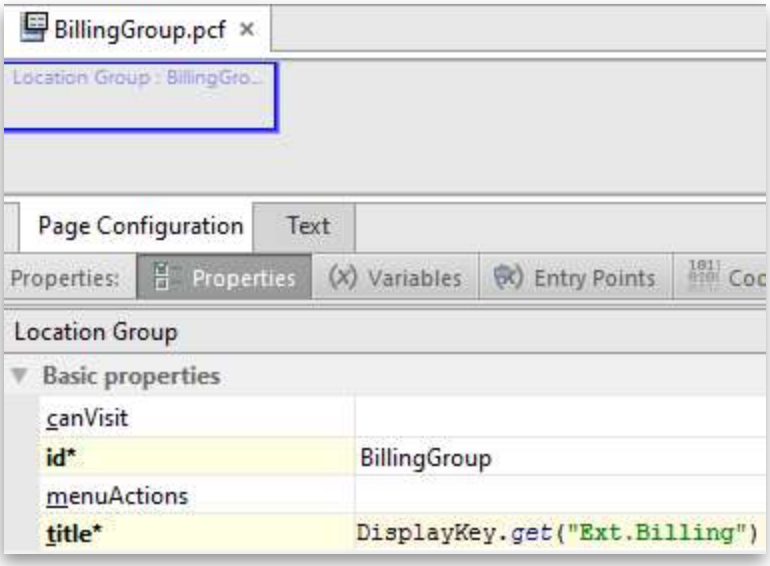
1. **Create a new pcf folder called training.**

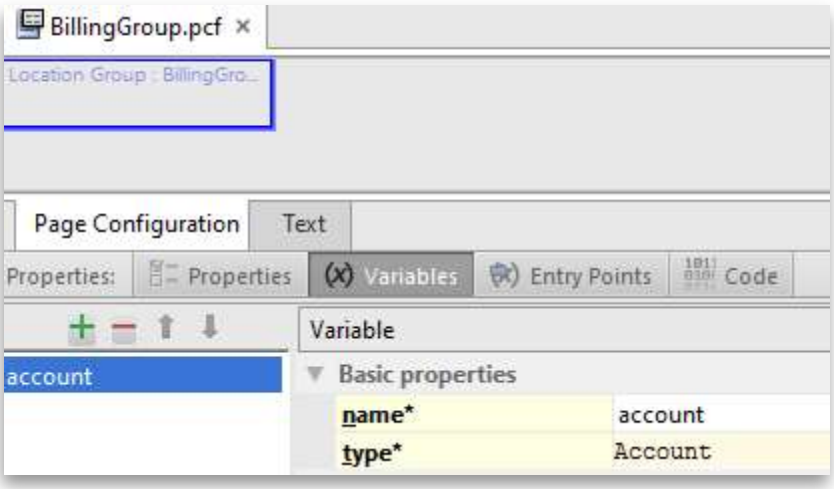


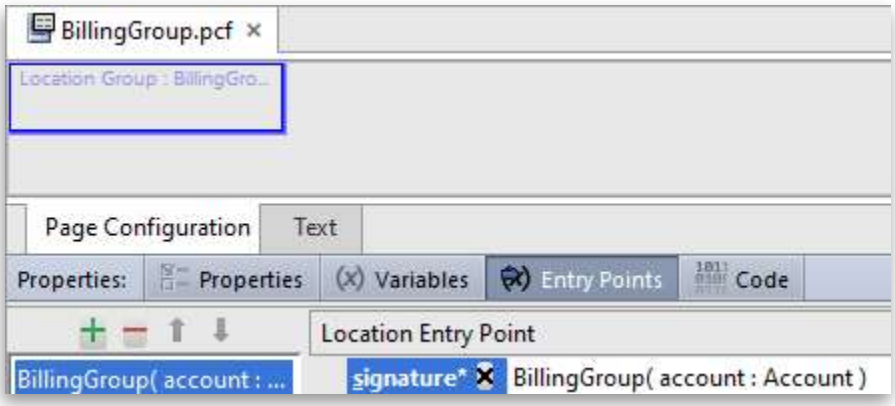

2. **Create a new location group pcf file called BillingGroup.**



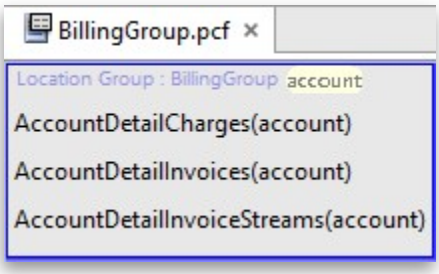

**3. Set the essential properties, entry point, and variables for BillingGroup pcf file.**



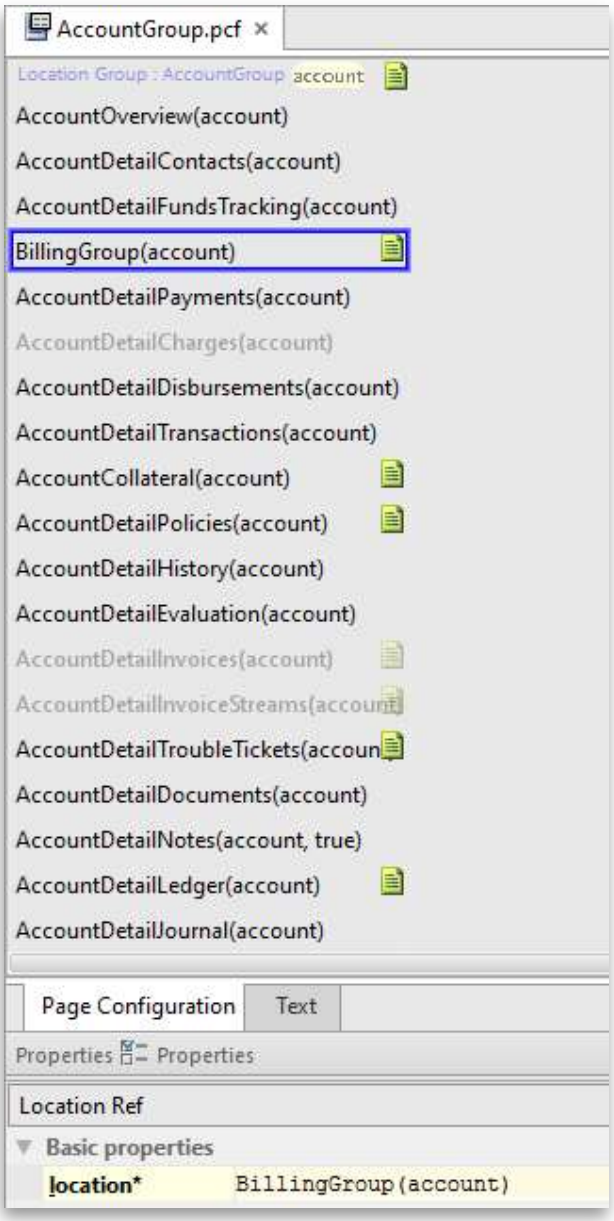



**4. Move the existing location ref for Charges, Invoices, and Invoice Streams to BillingGroup pcf file.**




**5. Add the BillingGroup location group to the account location group.**




6. **Deploy your changes by running Run → Reload Changed Classes.**
7. **Reload UI by running ALT+SHIFT+L prior to testing.**