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. **In Studio, select Run → Debugging Actions → 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 placement 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 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.
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 placement 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

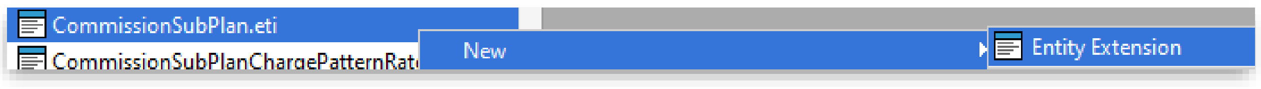
14. Go to the charges screen (Account tab → Charges) and click the placement date of the down payment invoice item for the premium charge.

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

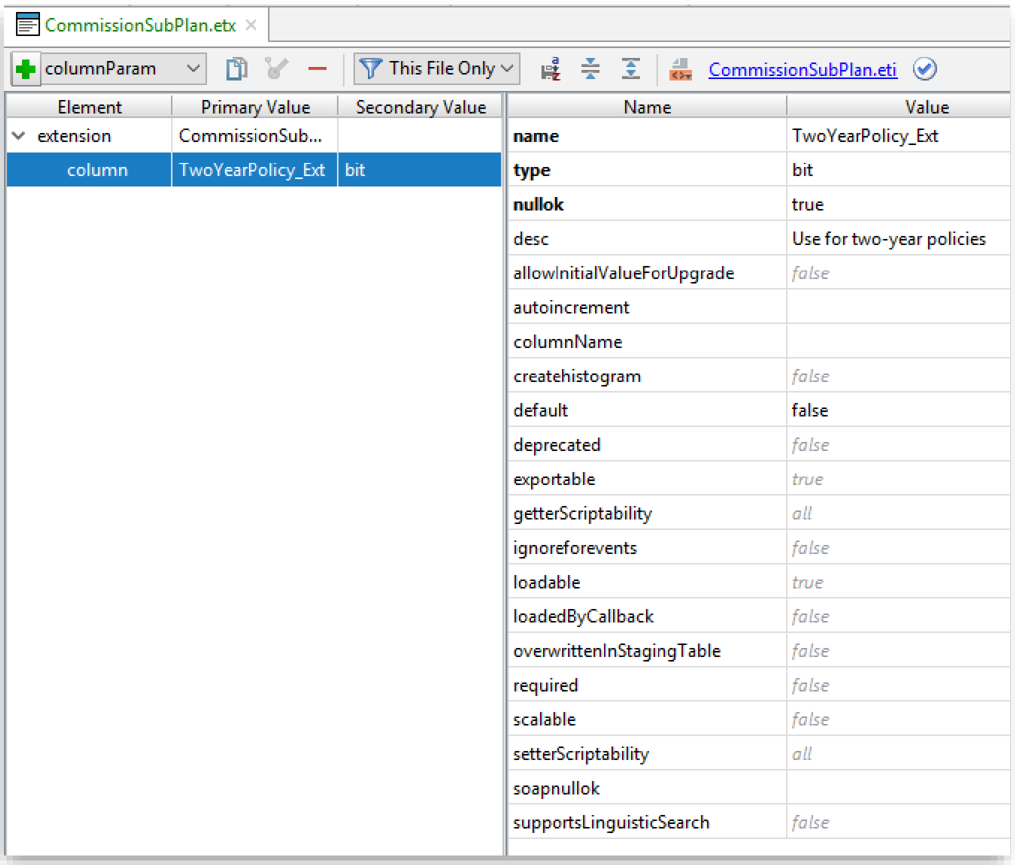
6.2.4 Solution

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.

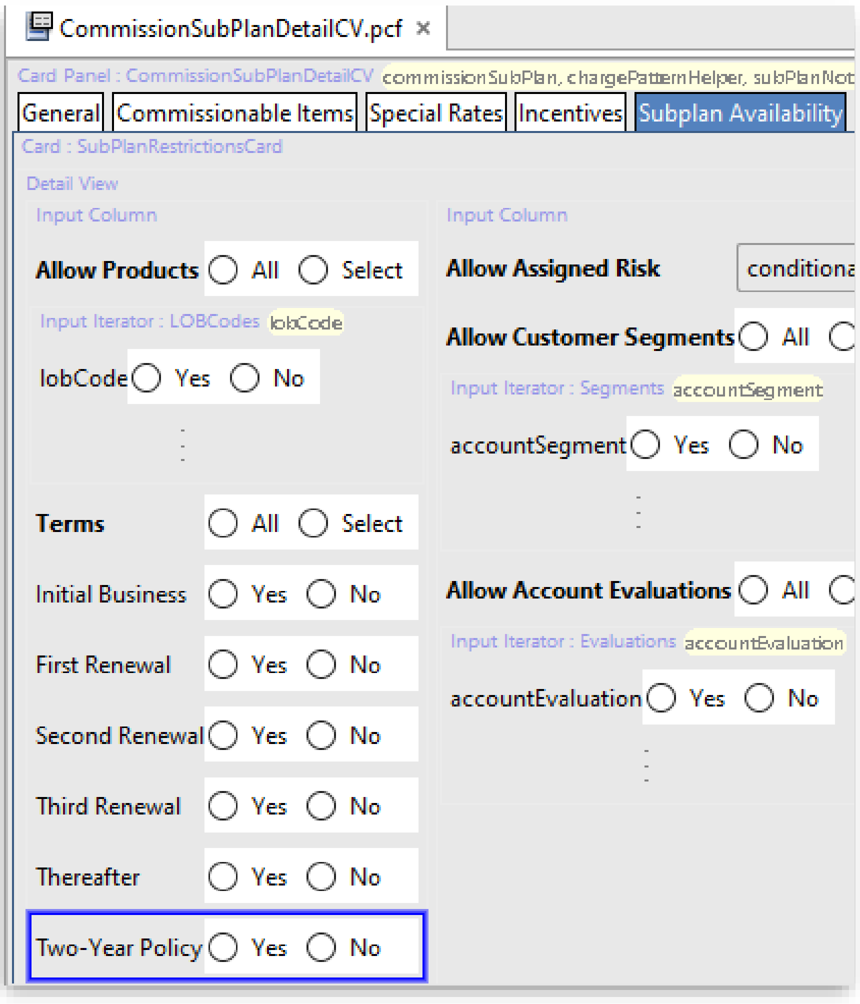

CommissionSubPlan.etx

columnParam | This File Only | CommissionSubPlan.eti

Element	Primary Value	Secondary Value
extension	CommissionSub...	
column	TwoYearPolicy_Ext	bit

Name	Value
name	TwoYearPolicy_Ext
type	bit
nullok	true
desc	Use for two-year policies
allowInitialValueForUpgrade	*false*
autoincrement	
columnName	
createhistogram	*false*
default	false
deprecated	*false*
exportable	*true*
getterScriptability	*all*
ignoreforevents	*false*
loadable	*true*
loadedByCallback	*false*
overwrittenInStagingTable	*false*
required	*false*
scalable	*false*
setterScriptability	*all*
soapnullok	
supportsLinguisticSearch	*false*

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 conditiona

Allow Customer Segments ◯ All ◯

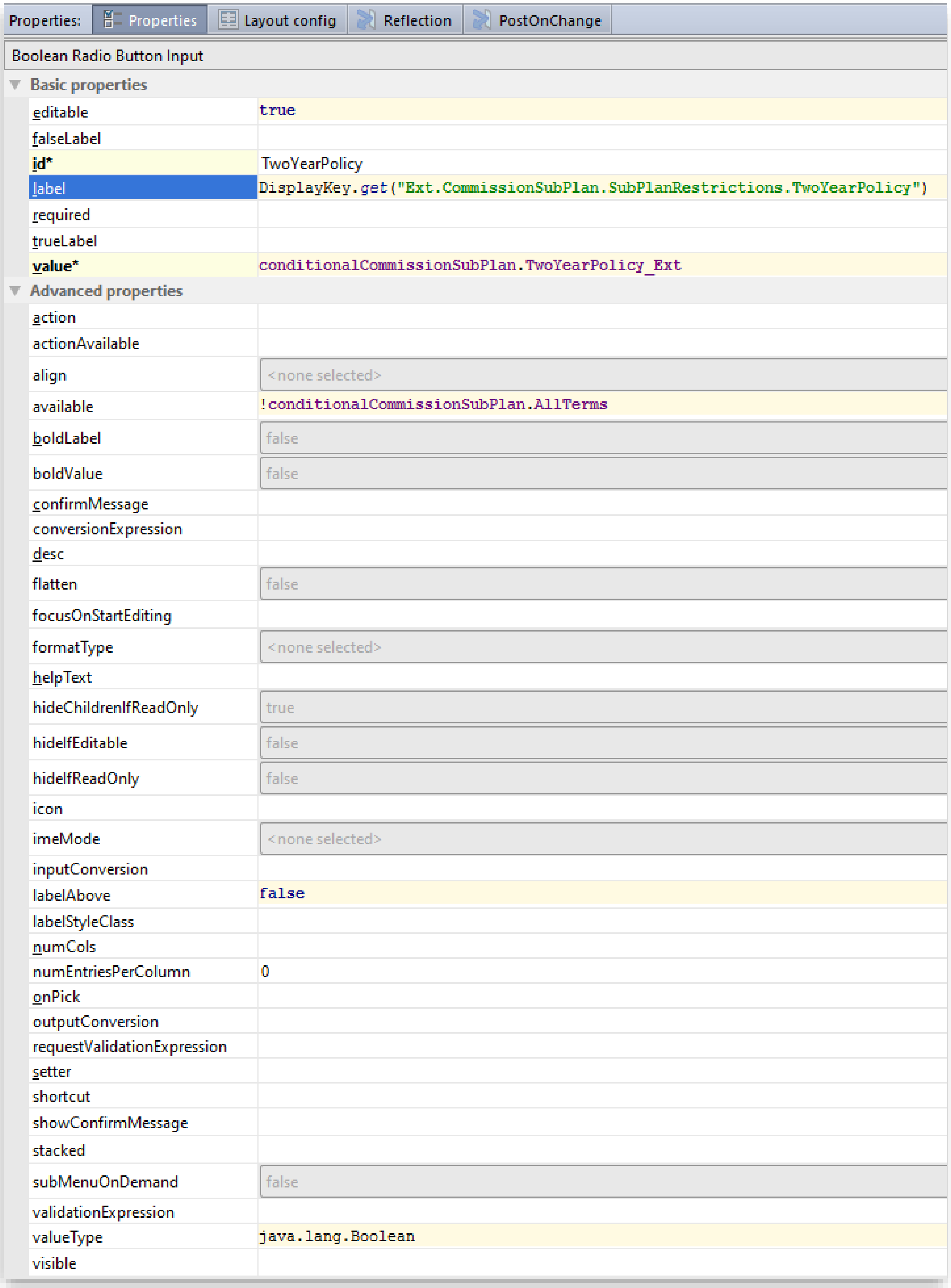
Input Iterator : Segments accountSegment

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. **In Studio, select Run → Debugging Actions → 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
  }
```