Lesson 1 Configuration Basics

1.1 Prerequisites

For this lab, use BillingCenter 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 Load sample data

1. In this lab, you practice the same steps that you performed during the demo. Here are the steps you need to complete:
2. Reset your database to an empty BillingCenter database. (HINT: Use Studio to drop your existing BillingCenter database and then start Server.)
3. Use Server Tools (Alt+Shift+T) to load the BillingCenter sample data. (Internal Tools → BC Sample Data)
4. How many accounts were created when the sample data was loaded?
5. Use the Quick Jump to view a list of the data builder methods that are available for an account. (Run Account ListAllMethods)
6. Use the Quick Jump to create a new account that has one policy and no producer. (Run Account with1PolicyWithNoProducer)
7. Reset your BillingCenter database again.
8. Use the Excel Data Loader to load the Admin, Plan, and Sample data from the configuration. Refer the Cookbook Recipe: Steps for using the Data Loader if needed.
9. Load the additional sample data from TrainingSampleConfigData.xls. This file is in the C:\GW\BillingCenter\modules\configuration\config\datafiles folder.

1.2.1 Solution

1. Reset your database to an empty BillingCenter database. (HINT: Use Studio to drop your existing BillingCenter database and then start Server.)
    a. Stop server if it is running
    b. In the Studio toolbar, change the 

dropdown to 

    c. From the toolbar, click the  to run the DropDB command.
    d. Once the DropDB command completes, change the dropdown back to Server and click the  to start server.
    e. Note: You may also use the Run file menu to most of these actions.
2. Use Server Tools (Alt+Shift+T) to load the BillingCenter sample data. (Internal Tools → BC Sample Data)
    a. How many accounts were created when the sample data was loaded?
        i. 2
3. Use the Quick Jump to view a list of the data builder methods that are available for an account. (Run Account ListAllMethods)
4. Use the Quick Jump to create a new account that has one policy and no producer. (Run Account with1PolicyWithNoProducer)

5. Reset your BillingCenter database again.
6. Use the Excel Data Loader to load the Admin, Plan, and Sample data from the configuration.
7. Load the additional sample data from TrainingSampleConfigData.xls. This file is in the C:\GW\BillingCenter\modules\configuration\config\datafiles folder.

1.3 Configure invoice fee behavior

Succeed Insurance uses variable invoice fees for some of their accounts. In this lab, you will implement changes to BillingCenter to support their requirements for variable invoice fees.

1.3.1 Requirements

Spec 1 Variable Invoice fees (on the account billing plan) are calculated as percentage rate of the invoice amount.

Spec 2 The invoice fee rate should be specified on the billing plan

Spec 3 A flag is needed on the billing plan to indicate if the default invoice fee should be overridden by the variable invoice fee

1.3.2 Tasks

1. Add new fields to the billing plan

Be sure to follow best practices, by following the data model extension naming conventions.

   a. Extend the BillingPlan entity
   b. Add a new percentagedec element to hold the variable invoice fee rate.
   c. Add a new bit element to indicate if the default invoice fee should be overridden by the variable invoice fee.
   d. Regenerate the data dictionary
   e. Deploy data model changes

2. Update the UI to display the new fields on the Billing Plan

Be sure to follow best practices, by creating display keys for the labels.

   a. Add the new bit element to the BillingPlanFeeHandlingInputSet.default.pcf. Set the label for this field to “Use Variable Rate Invoice Fee”.
   b. Add the new percentagedec element. Set the label for this field to “Variable Invoice Fee Rate”.

3. Modify the Fees and Thresholds plugin to implement the business requirements

Be sure to follow best practices, by commenting out the existing code.

   a. Open the Fees and Thresholds plugin.
   b. Update getInvoiceFeeAmountOverride ()method in FeesThresholds.gs plugin.
   c. Deploy code changes.

1.3.3 Testing procedure

1. Clone the BP01 billing plan.

Field	Value
Name	Variable Invoice Fee Test
Use Variable Rate Invoice Fee	Yes
Variable Invoice Fee Rate	0.25%

2. Manually create an account that uses your new billing plan.

Field	Value
Account Name	Invoice Fee Test
Billing Plan	Variable Invoice Fee Test
Select any value for remaining fields	

3. Add a policy to the account.

Field	Value
Policy #	IATT01
Payment Plan	PP02
Premium	$5000

4. Advance the system clock to the invoice date of the first invoice.
5. Run the invoice batch process.
6. In BillingCenter, go to the Account tab → Charges screen and confirm that one invoice fee of $1.25 was added to the billed invoice.

1.3.4 Solution

Solution

Exact details on how to complete the exercise.

1. Add new fields to the billing plan

   Be sure to follow best practices, by following the data model extension naming conventions.

   a. Extend the BillingPlan entity

   In Studio, go to configuration → config → Extensions → Entity. Then right click on Entity, select New → Entity Extension. In the Entity Extension window, enter “BillingPlan” and then click the OK button.

   b. Add a new percentagedec element to hold the variable invoice fee rate.

   Add a new column to the BillingPlan.etx. As listed below:

   Name: InvoiceFeeRate_Ext

   Type: percentagedec

   Nullok: true

   Default: 0

   c. Add a new bit element to indicate if the default invoice fee should be overridden by the variable invoice fee.

   Add a new column to the BillingPlan.etx. As listed below:

   Name: VariableInvoiceFee_Ext

   Type: bit

   Nullok: true

   Default: false

   d. Regenerate the data dictionary

   From a command line, execute the gwb genDataDictionary command.

   e. Deploy data model changes

   Restart server

2. Update the UI to display the new fields on the Billing Plan

   Be sure to follow best practices, by creating display keys for the labels.

   a. Add the new bit element to the BillingPlanFeeHandlingInputSet.default.pcf.. Set the label for this field to “Use Variable Rate Invoice Fee”.

   Add a new Boolean Radio Button Input, below the PlanMultiCurrencyFeeThresholdInputSet. Set the editable property to reference the planNotInUse variable. Set the id to UseVariableRateInvoiceFee. Set the label to

DisplayKey.get("Ext.UseVariableRateInvoiceFee"). Create a display key for this label. Set the value to billingPlan.VariableInvoiceFee_Ext.

b. Add the new percentagedec element. Set the label for this field to “Variable Invoice Fee Rate”.

Add a new Text Input, below the pcf element you just added. Set the editable property to reference the planNotInUse variable. Set the id to VariableInvoiceFeeRate. Set the label to DisplayKey.get("Ext.VariableInvoiceFeeRate"). Create a display key for this label. Set the value to billingPlan.InvoiceFeeRate_Ext. Set the valueType to java.math.BigDecimal.

3. Modify the Fees and Thresholds plugin to implement the business requirements

Be sure to follow best practices, by commenting out the existing code.

a. Open the Fees and Thresholds plugin.

In Studio, open the FeesThresholds.gs class.

b. Update getInvoiceFeeAmountOverride ()method in FeesThresholds.gs plugin.

Comment out the existing method and the add the following code to this class.

```
override function getInvoiceFeeAmountOverride(defaultAmount : MonetaryAmount,
                    invoice : AccountInvoice) : MonetaryAmount {
             return invoice.Account.BillingPlan.VariableInvoiceFee_Ext ?
                 invoice.Amount.multiply(invoice.Account.BillingPlan.InvoiceFeeRate_Ext / 100) : defaultAmount
}
```

c. Deploy code changes.

In Studio, select Run → Debugging Actions → Reload Changed Classes