Education

PolicyCenter Configuration Kickstart

Student Workbook

Table of Contents

Introduction

Welcome to the Guidewire PolicyCenter Configuration - **Kickstart** 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.

Sample Accounts

If the large sample data is loaded in PolicyCenter, it contains the following sample accounts for you to use to complete the exercises. It is recommended that you follow all the labs to create your own new accounts.

Account No.	Type	First Name	Last Name
C000143542	Person	Ray	Newton
B000457766	Person	Erica	Billings
C000456353	Person	Mary	Taylor
C000148456	Person	Bill	Kinman
A000377766	Person	Dave	Worthington

Account No.	Type	Company Name
C000212105	Company	Wright Construction
C000478975	Company	Big Lake Bakery
C000478567	Company	Calloway Cheese Factory
A000377655	Company	Dominion Logistics
B000467655	Company	Earth Tech

Open navigation pane

Tip

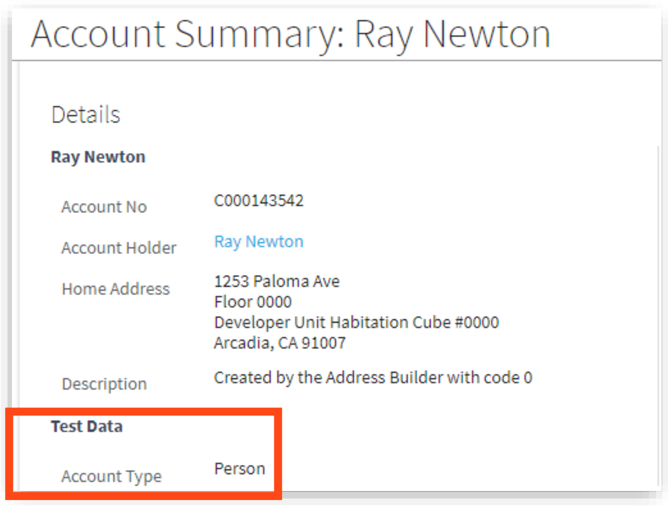
Open navigation pane in a PDF file

If you do not see the left navigation pane, you can turn it on. In a PDF file, click the bookmark icon on the left. Or click the Adobe Acrobat menu item **View → Show/Hide → Navigation Panes → Bookmarks**.

Lesson 1 PolicyCenter Data Model

1.1 Requirements

Configure PolicyCenter to meet the following customer requirement from Succeed Insurance. During the implementation of PolicyCenter, the implementation team for Succeed Insurance will be working with imported test data. When viewing any account, developers want to know if the account was created by a user or was imported test data. They also want to know quickly if the account is of type company or person.



Account Summary: Ray Newton

Details

Ray Newton

Account No	C000143542
Account Holder	Ray Newton
Home Address	1253 Paloma Ave Floor 0000 Developer Unit Habitation Cube #0000 Arcadia, CA 91007
Description	Created by the Address Builder with code 0

Test Data

Account Type	Person

Implementation Notes

The PolicyCenter implementation team has provided additional notes:

- All the test data has an Account Number which starts with the characters “A”, “B”, or “C”. Therefore, to determine if an account is a test account or not, evaluate the first character of the Account Number. You can use the startsWith string method available on string fields and it has the following syntax:

```
Object.fieldname.startsWith(stringToCheck)
```

- If a piece of logic is used in multiple places, it is better to declare it as a function. Therefore, you should create an enhancement called `isTestData_Ext()` which evaluates if an account has been created from the loaded test data.
- The account type is determined by the account holder’s type. Therefore, you can check the account holder contact’s subtype.

1.2 Configuration

Activity

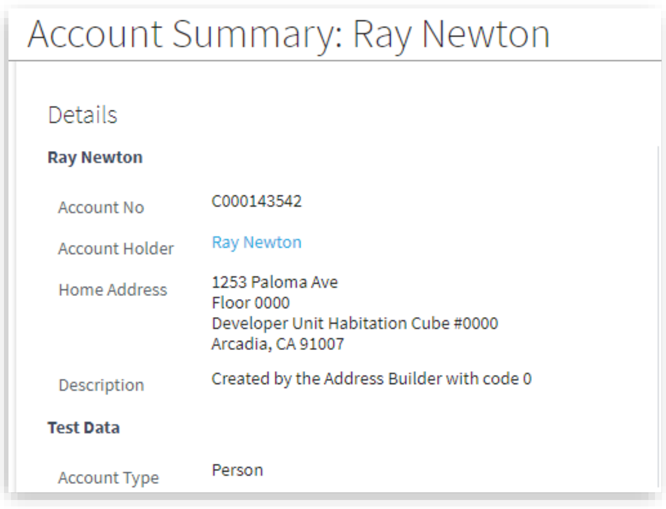
1. **Add an enhancement function to the `Account` entity to check if the account is test data.**
2. **Add a label and a field to the Account Summary detail view, which is displayed at the top of the Summary screen when viewing an account.**
   a) Add a label with the following attributes:
      - The label appears below the account description field.
      - The label is "`Test Data`".
      - It appears only if the account is a test data account.

   b) Add a field with the following attributes:
      - It appears below the Test Date label.
      - The field is not editable.
      - The label for the field is "`Account Type".`
      - It displays the value of the Account Holder Contact's type.
      - It always appears.
3. **Restart the server.**

1.3 Verification

1. **Go to the PolicyCenter application.**
2. **Log in as Alice Applegate.**
3. **Navigate to an account loaded from the sample data. Verify that the `Test Data` label and the `Account Type` field are displayed.**

   These accounts typically start with a letter. Account C000143542 is a sample data account and is person type, while Account C000212105 is company type. Refer to the section Introduction / Sample Accounts for more details.



Account Summary: Ray Newton

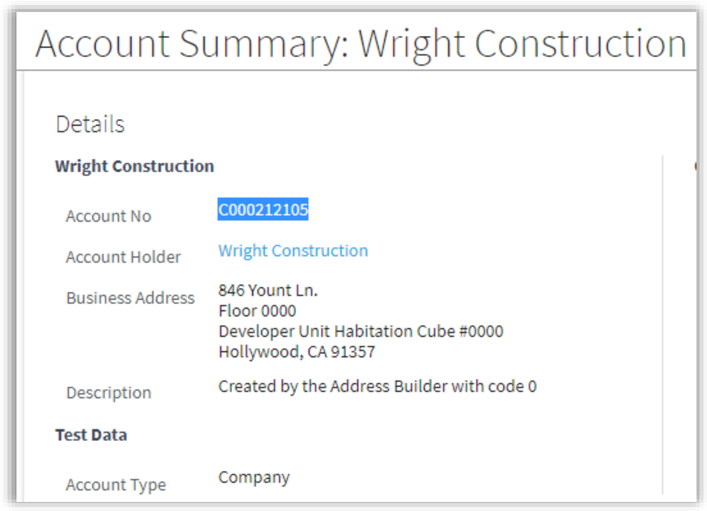
Details

Ray Newton

Account No	C000143542
Account Holder	Ray Newton
Home Address	1253 Paloma Ave Floor 0000 Developer Unit Habitation Cube #0000 Arcadia, CA 91007
Description	Created by the Address Builder with code 0

Test Data

Account Type	Person



Account Summary: Wright Construction

Details

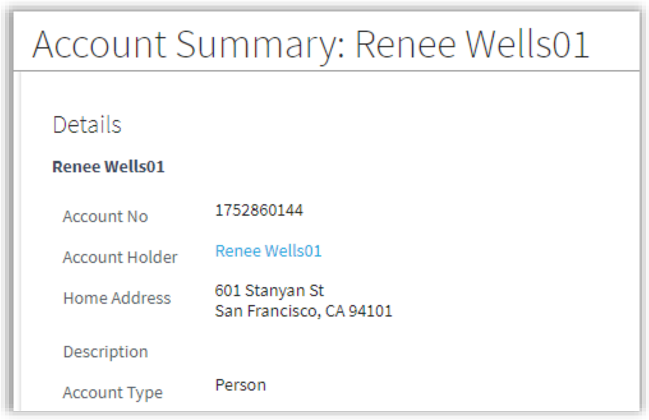
Wright Construction

Account No	C000212105
Account Holder	Wright Construction
Business Address	846 Yount Ln. Floor 0000 Developer Unit Habitation Cube #0000 Hollywood, CA 91357
Description	Created by the Address Builder with code 0

Test Data

Account Type	Company

4. **Navigate to an account you created previously. Or create a new account using the new account wizard. Choose producer as Archie Armstrong. Fill in the other details.**
5. **Once the account is available, verify that the `Test Data` label does not appear, but the account type field appears.**


Account Summary: Renee Wells01

Details

Renee Wells01

Account No	1752860144
Account Holder	Renee Wells01
Home Address	601 Stanyan St San Francisco, CA 94101
Description	
Account Type	Person

Stop

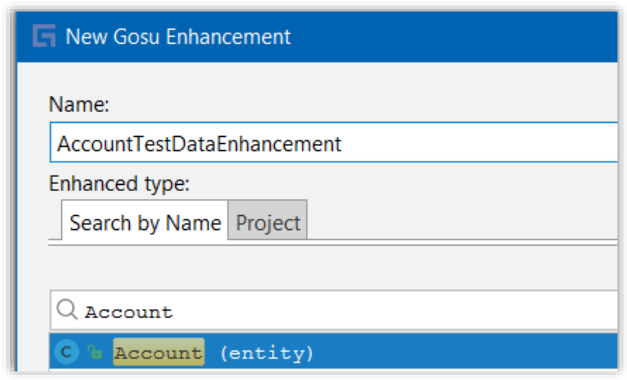
1.4 Solutions

Solution

1. **Add an enhancement function to the `Account` entity to check if the account is test data.**
   a) In Studio, go to `configuration/gsrc`.
   b) Right click on `gsrc` and select **New → Package** for Succeed Insurance entity enhancements.

      `si.pc.enhancement.entity`

   c) Right click on the new package and select **New → Gosu Enhancement.**
   d) Create `AccountTestDataEnhancement` that enhances the `Account` entity.



Name: AccountTestDataEnhancement

Account

Account (entity)

e) Add a function `isTestData_Ext()`.

```
package si.pc.enhancement.entity

enhancement AccountTestDataEnhancement : Account {
  function isTestData_Ext(): boolean {
    return this.AccountNumber.startsWith("A") or
        this.AccountNumber.startsWith("B") or
        this.AccountNumber.startsWith("C")
  }

}
```

2. Add a label and a field to the Account Summary detail view.

a) Press `CTRL + n` to open the file `AccountDetailsDetailViewTile.pcf` under config/web/pcf/account/overview.

b) Add a new label below the field Description.

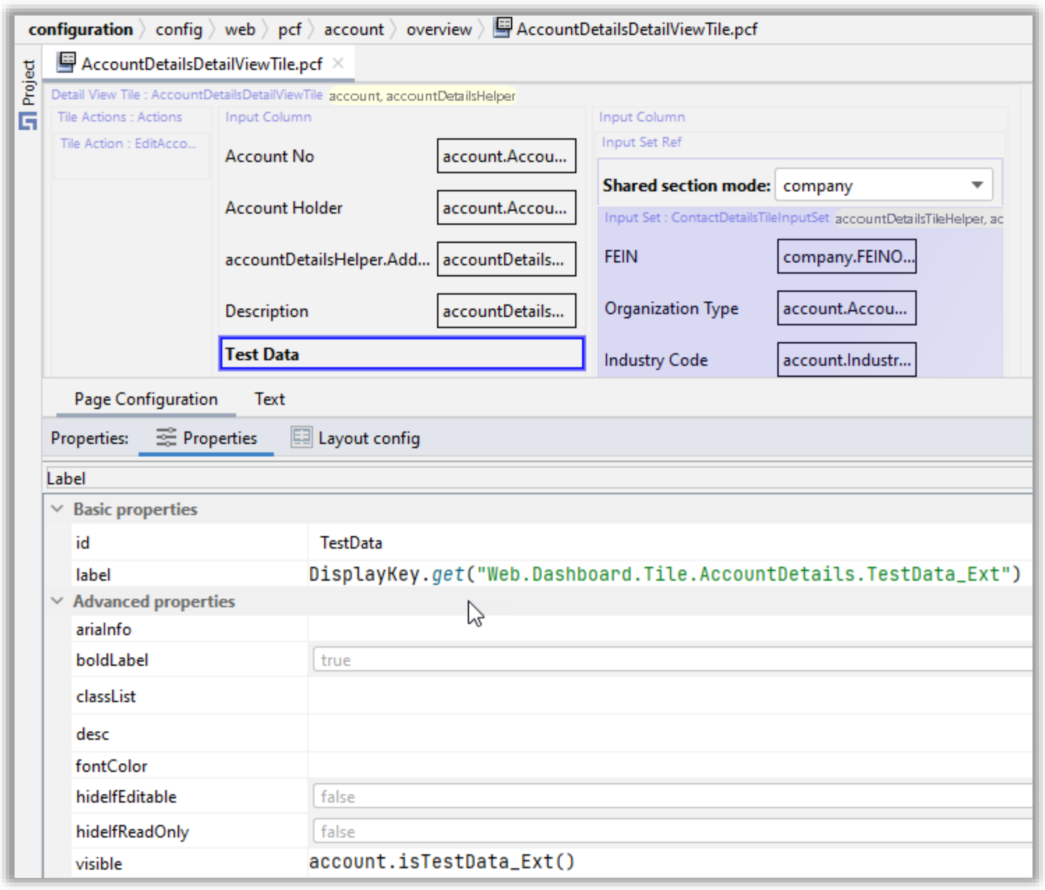
c) Enter details such as ID, label, visible, etc.

```
Label = DisplayKey.get("Web.Dashboard.Tile.AccountDetails.TestData_Ext")

Visible = account.isTestData_Ext()
```

d) If the display key does not exist, put the mouse cursor on the display key and press ALT + Enter to create it.

```
Web.Dashboard.Tile.AccountDetails.TestData_Ext = Test Data
```


configuration 〉 config 〉 web 〉 pcf 〉 account 〉 overview 〉 AccountDetailsDetailViewTile.pcf

AccountDetailsDetailViewTile.pcf

Detail View Tile : AccountDetailsDetailViewTile account, accountDetailsHelper

Account No	account.Accou...
Account Holder	account.Accou...
accountDetailsHelper.Add...	accountDetails...
Description	accountDetails...
Test Data	

Shared section mode: company

Input Set : ContactDetailsTileInputSet accountDetailsTileHelper, ac

FEIN	company.FEINO...
Organization Type	account.Accou...
Industry Code	account.Industr...

Label

Basic properties

id	TestData
label	`DisplayKey.get("Web.Dashboard.Tile.AccountDetails.TestData_Ext")`

Advanced properties

arialnfo	
boldLabel	true
classList	
desc	
fontColor	
hideIfEditable	false
hideIfReadOnly	false
visible	`account.isTestData_Ext()`

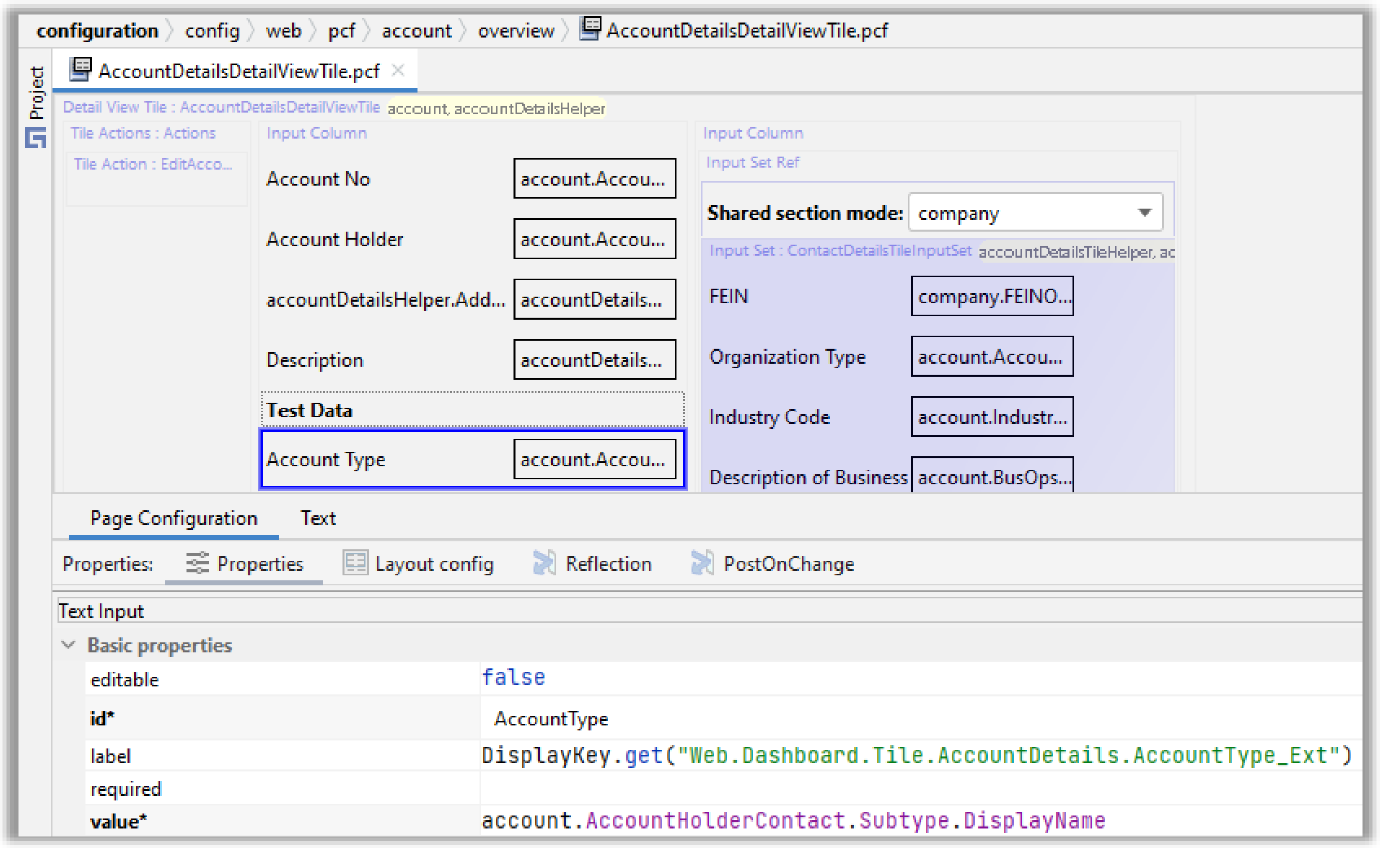
e) Add a Text Input element below the element `Test Data` to display the account holder contact type.

Text Input property:

```
value = account.AccountHolderContact.Subtype.DisplayName
label = DisplayKey.get("Web.Dashboard.Tile.AccountDetails.AccountType_Ext")
```

DisplayKey:

```
Web.Dashboard.Tile.AccountDetails.AccountType_Ext = Account Type
```



configuration 〉 config 〉 web 〉 pcf 〉 account 〉 overview 〉 AccountDetailsDetailViewTile.pcf

Detail View Tile : AccountDetailsDetailViewTile account, accountDetailsHelper

Label	Value
Account No	account.Accou...
Account Holder	account.Accou...
accountDetailsHelper.Add...	accountDetails...
Description	accountDetails...
Test Data	
Account Type	account.Accou...

Shared section mode: company

Input Set : ContactDetailsTileInputSet accountDetailsTileHelper, ac

Label	Value
FEIN	company.FEINO...
Organization Type	account.Accou...
Industry Code	account.Industr...
Description of Business	account.BusOps...

Text Input

Basic properties

Property	Value
editable	false
id*	AccountType
label	DisplayKey.get("Web.Dashboard.Tile.AccountDetails.AccountType_Ext")
required	
value*	account.AccountHolderContact.Subtype.DisplayName

3. Restart the server.

1.5 References

Review

PolicyCenter Data Model

There are two primary entities in the PolicyCenter data model: `Policy` and `Account`. Including these two entities, there are about 25 major entities in the PolicyCenter data model. These entities can be broken down into five groups. See colored groups below. Each group can be thought of as answering a business question:

- What does each policy period look like? (Charcoal)
- What is covered on the policy? (coverables and coverages) (Burgundy)
- Who holds the policy? (account contacts and locations) (Brown)
- What is the work to be done for the policy? (activities and workflows) (Orange)
- Who is responsible for processing the policy? (the users and groups working for the carrier on the policy) (Blue)

Lesson 2 Location Groups and Pages

2.1 Requirements

In this exercise, you add a new left menu link for policy terms to the Account File in PolicyCenter. Clicking the menu link opens a new page that displays a list of all the policy terms.

2.2 Configuration

Activity

1. **Add a new Location Ref for Policy Terms in the Account File Location Group.**
2. **Create a new page for Policy Terms. The following are some tips.**
   a) You can copy an existing page, for example `AccountFile_Roles.pcf`.

      Note: It is recommended not to copy and reuse any portion on the `AccountFile_Summary` page because it uses the dashboard and tile elements, which are special for summary pages.

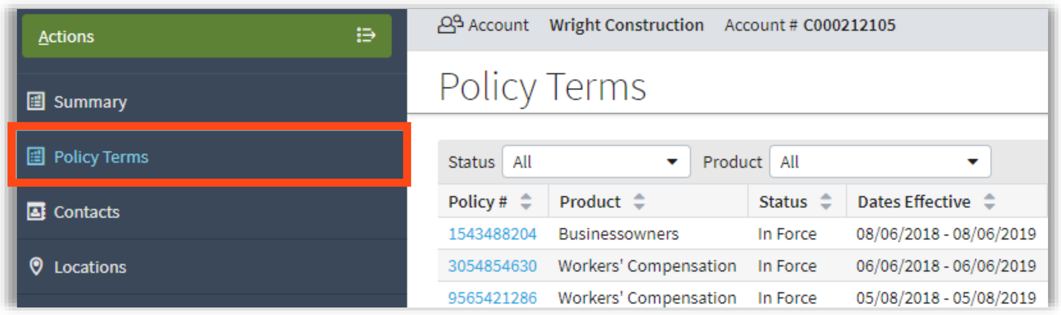
   b) Update the titles, entry points, etc.

   c) Remove the `Screen` element.

   d) Copy the `ScreenRef` element from `Policy_CurrentPoliciesPopup.pcf` and paste it into the new page for Policy Terms.

2.3 Verification

1. **Log in to PolicyCenter as Alice Applegate with aapplegate/gw.**
2. **Reload PCF files (Shift + Alt + L).**
3. **Go to any account.**
4. **Policy Terms link on sidebar displays the Policies list view.s**



Policy Terms

Status: All   Product: All

Policy #	Product	Status	Dates Effective
1543488204	Businessowners	In Force	08/06/2018 - 08/06/2019
3054854630	Workers' Compensation	In Force	06/06/2018 - 06/06/2019
9565421286	Workers' Compensation	In Force	05/08/2018 - 05/08/2019

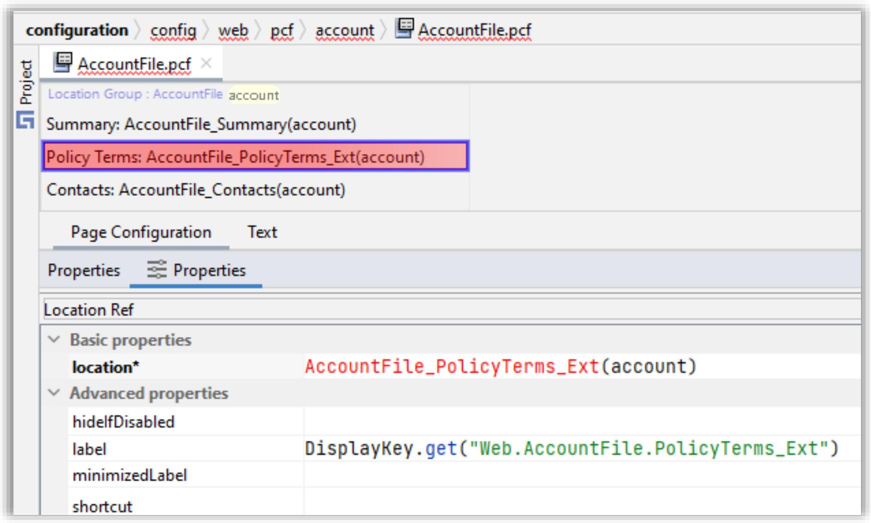
Stop

2.4 Solutions

Solution

1. **Add a location ref for Policy Terms to the location group file `AccountFile.pcf`.**

   Note: The errors (red areas around the new location ref) will be resolved when the following steps are completed.


configuration 〉 config 〉 web 〉 pcf 〉 account 〉 AccountFile.pcf
Project
AccountFile.pcf
Location Group : AccountFile account

Summary: AccountFile_Summary(account)
Policy Terms: AccountFile_PolicyTerms_Ext(account)
Contacts: AccountFile_Contacts(account)

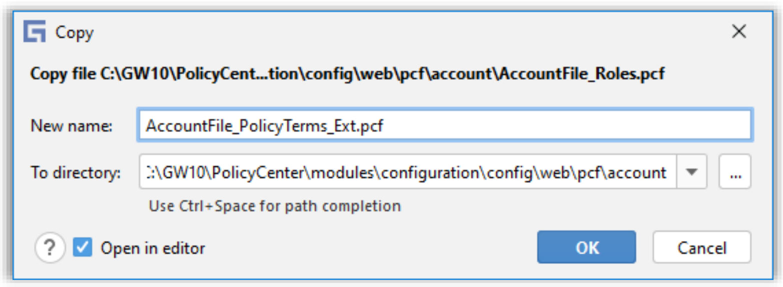
Page Configuration
Text
Properties
Properties

Location Ref

Basic properties	
location*	AccountFile_PolicyTerms_Ext(account)
Advanced properties	
hideIfDisabled	
label	DisplayKey.get("Web.AccountFile.PolicyTerms_Ext")
minimizedLabel	
shortcut	

2. **Create a new page for Policy Terms.**

   a) Copy `AccountFile_Roles.pcf` and paste it in the same folder. Name it `AccountFile_PolicyTerms_Ext.pcf`.



Copy

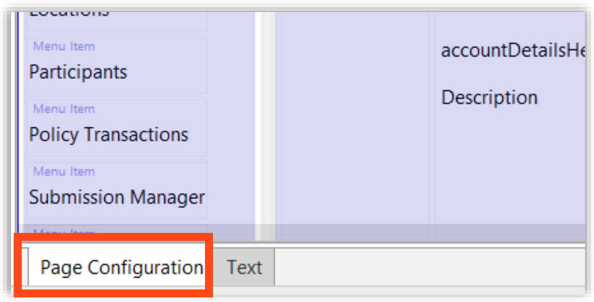
Copy file C:\GW10\PolicyCent...tion\config\web\pcf\account\AccountFile_Roles.pcf

New name: AccountFile_PolicyTerms_Ext.pcf

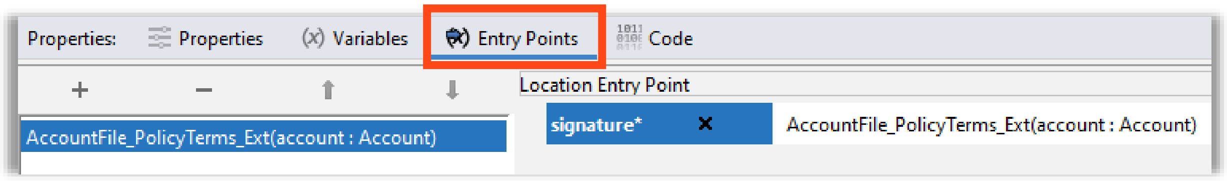
To directory: :\GW10\PolicyCenter\modules\configuration\config\web\pcf\account

Use Ctrl+Space for path completion

b) Click on `Page configuration` to open the PCF editor.



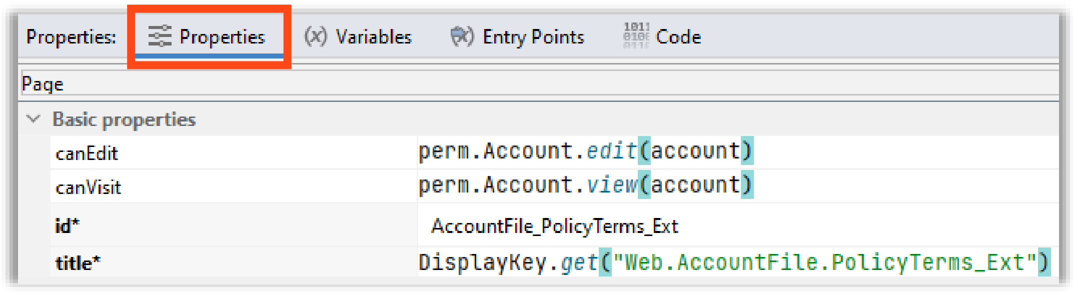
c) Click on the top level `Page` element and update the `Entry Points` to be `AccountFile_PolicyTerms_Ext(account: Account).`



AccountFile_PolicyTerms_Ext(account : Account)

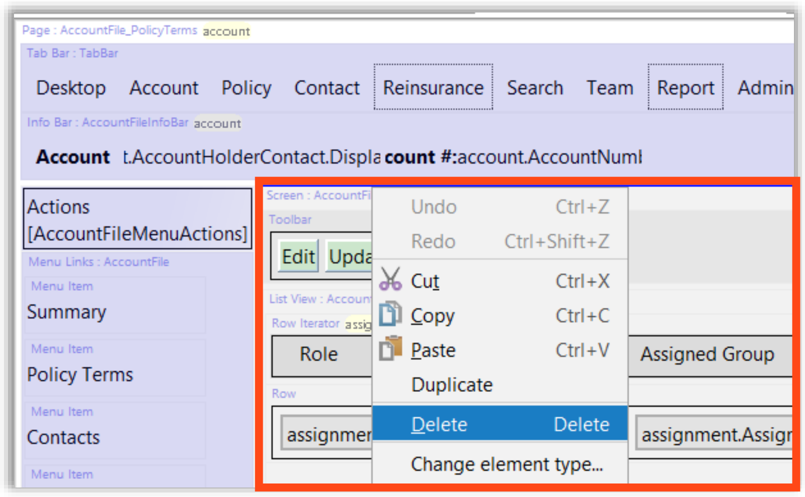
signature* AccountFile_PolicyTerms_Ext(account : Account)

d) Update the permissions and page title on the `properties` tab of the page element.

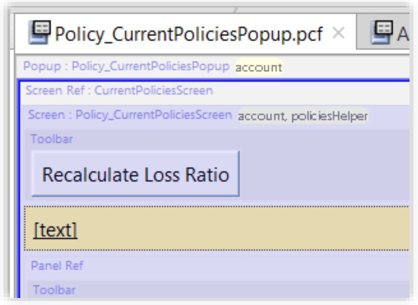


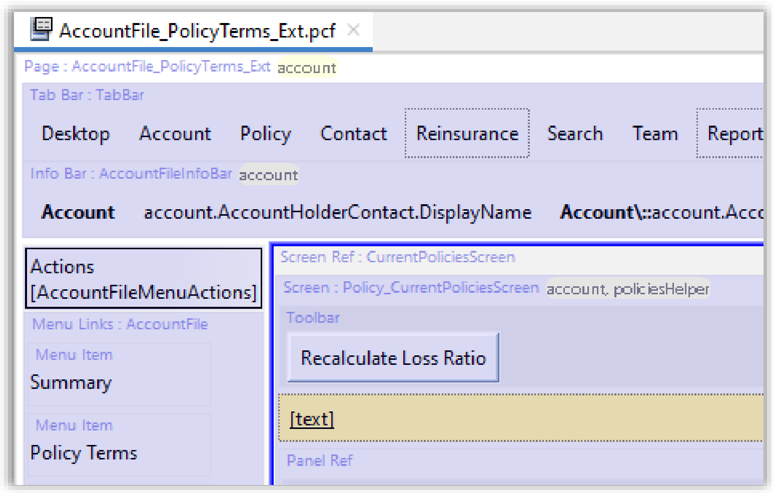
canEdit	perm.Account.edit(account)
canVisit	perm.Account.view(account)
id*	AccountFile_PolicyTerms_Ext
title*	DisplayKey.get("Web.AccountFile.PolicyTerms_Ext")

e) Select the `Screen` element and delete it from the new PCF file.



f) Click CTRL + N and open the file `Policy_CurrentPoliciesPopup.pcf`. Copy the `ScreenRef` element in this file and paste it into `AccountFile_PolicyTerms_Ext.pcf`.


Policy_CurrentPoliciesPopup.pcf
A
Popup : Policy_CurrentPoliciesPopup account
Screen Ref : CurrentPoliciesScreen
Screen : Policy_CurrentPoliciesScreen account, policiesHelper
Toolbar
Recalculate Loss Ratio
[text]
Panel Ref
Toolbar


AccountFile_PolicyTerms_Ext.pcf
Page : AccountFile_PolicyTerms_Ext account
Tab Bar : TabBar
Desktop Account Policy Contact Reinsurance Search Team Report
Info Bar : AccountFileInfoBar account
Account account.AccountHolderContact.DisplayName Account\::account.Acco
Actions [AccountFileMenuActions]
Menu Links : AccountFile
Menu Item
Summary
Menu Item
Policy Terms
Screen Ref : CurrentPoliciesScreen
Screen : Policy_CurrentPoliciesScreen account, policiesHelper
Toolbar
Recalculate Loss Ratio
[text]
Panel Ref

g) Go to the PolicyCenter application and reload the pages to verify the changes (Shift + Alt + L).

2.5 References

Cookbook Recipe

Steps to configure a location group

1. **Create new PCF of type Location Group**
2. **Set essential information:**
   a) title
   b) Entry point with matching variable (Entry Points tab)

      Syntax: `EntryPointName(ObjectName: ObjectType)`

   c) A variable for each object in entry point (Variables tab)
3. **Add location references for all included pages and location groups**
4. **Call the new location group in the action property of a navigation widget**

   Syntax: `EntryPointName.go(variablesToPass)`

5. **Deploy changes**

Cookbook Recipe

Steps to configure a page

1. **Create new PCF of type Page**
2. **Set essential properties:**
   a) title
   b) Entry point with matching variable (Entry Points tab)
   c) A variable for each object in entry point (Variables tab)
   d) Add a screen
   e) Include panels for viewable and/or editable content
3. **For editable pages:**
   a) Set canEdit property
   b) Add EditButtons to the page's screen
4. **In the location group that will contain the page, add a locationref widget for navigation to the page**

   ```
   LocationPropertyOfLocationRef: EntryPointOfPage(objectsToPass)
   ```

5. **Deploy changes**

Review

Location Groups

A **location group** is a collection of locations used to provide structure and navigation for a group of related pages through menus or other interface elements.

Navigation Hierarchies:

- First level: tab bar
  - Clicking tab bar or menu item results in different location group.
- Second level: menu links (location refs)
  - Clicking menu link stays within the same location group.
  - Only the page that the menu link points to is changed.
  - The menu actions, menu links, info bar and tab bar are unchanged.

Information in a location group include:

- The **Entry points** and **variables** used by navigation widgets.
- The **Location refs** that point to pages or child location groups.
- The **menuActions** property specifies the file to use for contents of the Actions menu. It is optional.
- The **infoBar** property specifies the file to use for contents of the info bar. It is optional.
- The **tabBar** property specifies the file to use for the contents of the tab bar. It is optional.
- Ctrl + Click on the value for the menuActions, infoBar or tabBar opens the file link.
- The **canVisit** attribute determines whether a user can visit the location group. If blank, the canVisit attribute evaluates to true. When the attribute evaluates to false, the tab is not rendered in the user interface.
- Single-screen locations and wizards also have a **canEdit** attribute. Location groups do not have the canEdit property. Editability for location groups is controlled at the level of each page.

Review

Pages

A **page** is a location that contains a single screen in the main frame.

- Used exclusively within location groups.
- Menu link / location ref (in sidebar) pointing to a page.

```
LocationPropertyOfLocationRef: EntryPointOfPage(objectsToPass)
```

- The rendering of the page in Studio can look as if all the embedded elements are visible in the page.

Information in a page includes:

- Title label of a page in the user interface comes from page's title property.
- The **entry points** and **variables** used by location refs' location property.
- If **canVisit** evaluates to true, the link to a page is visible and clickable.
- A page is either in read-only or edit mode.
- If **canEdit** evaluates to true, the page has Edit/Update buttons.
- If **startInEditMode** is true, then page renders in Edit mode.

Lesson 3 Job Wizards

3.1 Requirements

In this exercise, you add a new jobwizardstep to the Submission JobWizard. The new step’s name is Producer Details. It contains the same information as the Producer of Record section from the Policy Info screen. It is located after Policy Info.

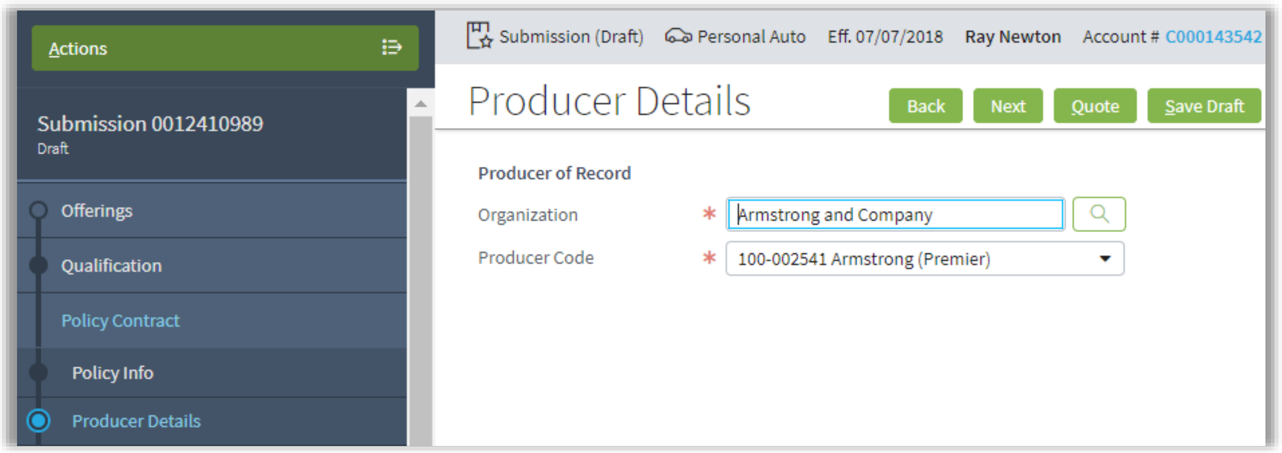
3.2 Configuration

Activity

1. **Create a new screen for Producer of Records.**
2. **Add a new JobWizardStep Producer Details in the Submission Wizard and point it to the new screen.**

3.3 Verification

1. **Log in to PolicyCenter as Alice Applegate, aapplegate/gw.**
2. **Reload PCF files.**
3. **Open a submission in draft mode or create a new submission.**
4. **The Producer Details step should be added after Policy Info and shows the Producer of Record information on the screen when selected.**



Producer Details

Producer of Record

Organization *	Armstrong and Company
Producer Code *	100-002541 Armstrong (Premier)

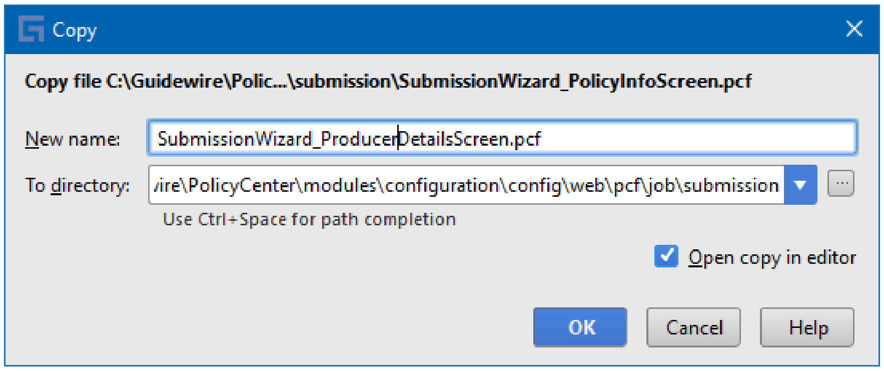
Stop

3.4 Solutions

Solution

1. Create a new screen for Producer of Record.

Copy `SubmissionWizard_PolicyInfoScreen.pcf` and name it `SubmissionWizard_ProducerDetailsScreen.pcf`



Copy file C:\Guidewire\Polic...\submission\SubmissionWizard_PolicyInfoScreen.pcf

New name: SubmissionWizard_ProducerDetailsScreen.pcf

To directory: /ire\PolicyCenter\modules\configuration\config\web\pcf\job\submission

Use Ctrl+Space for path completion

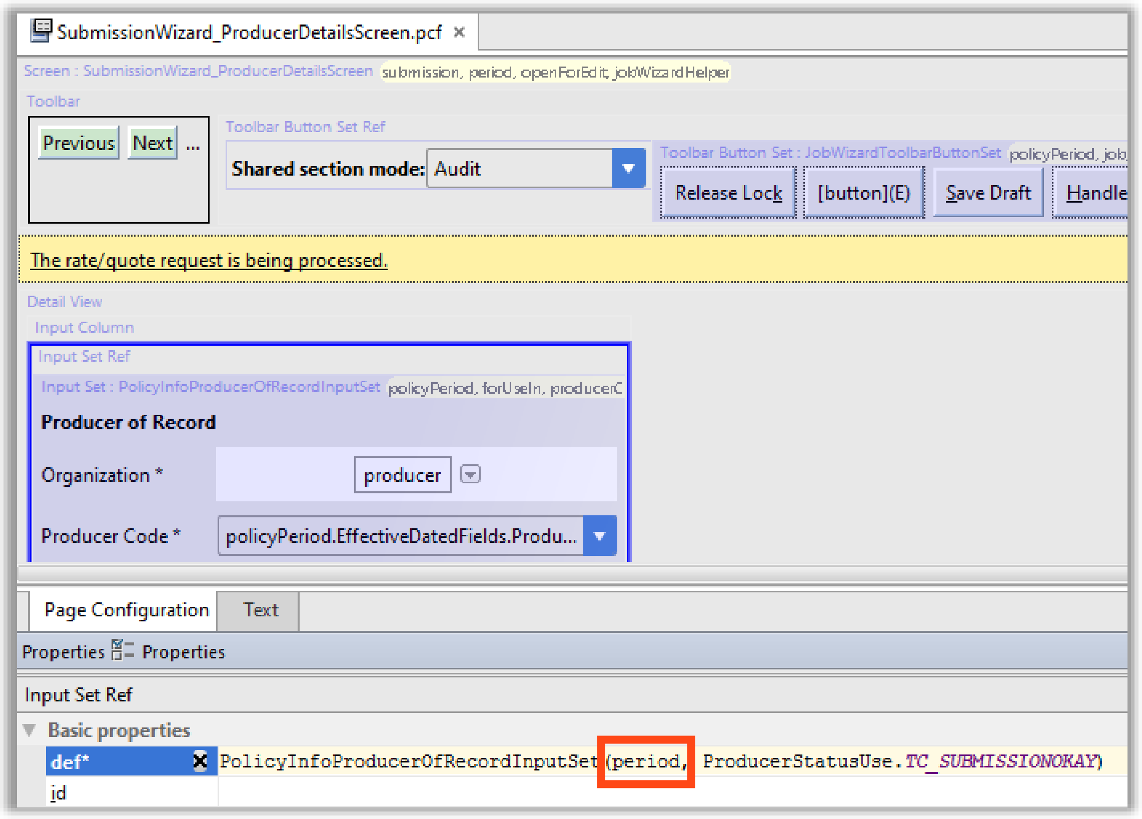
☑ Open copy in editor

2. Create content in the new screen.

a) In SubmissionWizard_ProducerDetailsScreen.pcf, delete every component except the toolbar.

b) Copy and paste the `InputSetRef` for Producer of Record from `SubmissionWizard_PolicyInfoDV.pcf` to `SubmissionWizard_ProducerDetailsScreen.pcf`.

c) You'll need to rename the parameters in the `InputSetRef` to match variables in the new screen. For example, change policyPeriod to period.


SubmissionWizard_ProducerDetailsScreen.pcf
Screen : SubmissionWizard_ProducerDetailsScreen submission, period, openForEdit, jobWizardHelper
Toolbar
Previous
Next
Toolbar Button Set Ref
Shared section mode: Audit
Toolbar Button Set : JobWizardToolbarButtonSet policyPeriod, job
Release Lock
[button](E)
Save Draft
Handle
The rate/quote request is being processed.
Detail View
Input Column
Input Set Ref
Input Set : PolicyInfoProducerOfRecordInputSet policyPeriod, forUseIn, producerC
Producer of Record
Organization *
producer
Producer Code *
policyPeriod.EffectiveDatedFields.Produ...
Page Configuration
Text
Properties
Properties
Input Set Ref
Basic properties
def*
id

```
PolicyInfoProducerOfRecordInputSet(period, ProducerStatusUse.TC_SUBMISSIONOKAY)
```

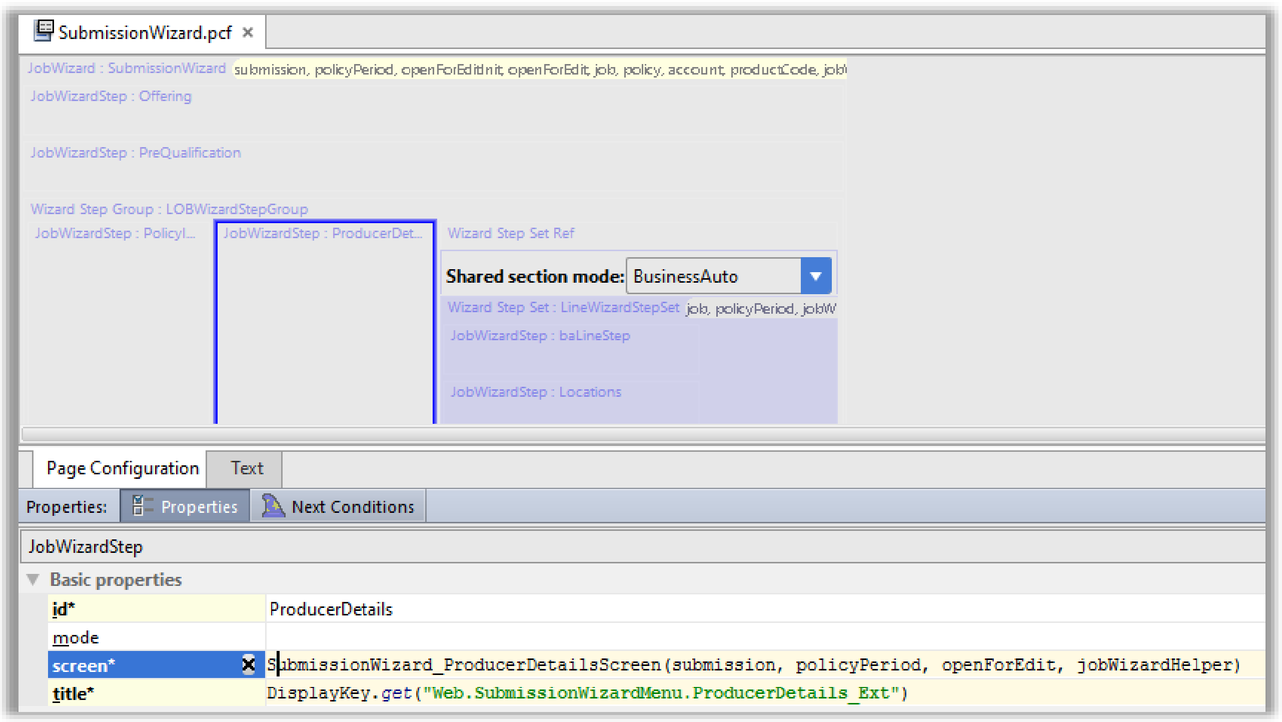
3. Add a new `JobWizardStep` for Producer Details in `SubmissionWizard.pcf`.

a) Duplicate the PolicyInfo step

b) Change the id field to ProducerDetails

c) Change the screen attribute and point it to the new Producer Details screen.

d) Create display keys for the screen's title.


SubmissionWizard.pcf
JobWizard : SubmissionWizard
submission, policyPeriod, openForEditInit, openForEdit, job, policy, account, productCode, jobW
JobWizardStep : Offering
JobWizardStep : PreQualification
Wizard Step Group : LOBWizardStepGroup
JobWizardStep : PolicyI...
JobWizardStep : ProducerDet...
Wizard Step Set Ref
Shared section mode: BusinessAuto
Wizard Step Set : LineWizardStepSet
job, policyPeriod, jobW
JobWizardStep : baLineStep
JobWizardStep : Locations
Page Configuration
Text
Properties:
Properties
Next Conditions
JobWizardStep
Basic properties

id*	ProducerDetails
mode	
screen*	SubmissionWizard_ProducerDetailsScreen(submission, policyPeriod, openForEdit, jobWizardHelper)
title*	DisplayKey.get("Web.SubmissionWizardMenu.ProducerDetails_Ext")

4. In the PolicyCenter application, press `Alt + Shift + L` to reload changes to pcf.

3.5 References

Reference

Job Wizard

A **job wizard** has job wizard steps that individually points to a screen and therefore the screens in a job wizard may have an order. Only one screen is active at a time in a job wizard.

The screens in a job wizard have an order (though it may be possible for a user to traverse the screens out of order). Job wizards collect information and perform operations while guiding the user through a complex business transaction. The UI functionality of job wizards is captured in a PCF. The actual Job process is usually defined in a Gosu class which coordinates the policy transaction and the job wizards interacts with. So, `SubmissionWizard` is a PCF, but `SubmissionProcess` is a Gosu class.

Information in a job wizard includes:

- Actions menu.
- Dependent steps, which are an ordered set of labels, each of which represents a single job wizard step and maps to a single screen. Some steps may not be accessible until a given step is complete.
- Independent steps, which are a set of labels, each of which maps to a single screen. Independent steps do not have a logical order and can be accessed at any time, regardless of what steps have been completed in the job wizard.
- Tab bar, which is the set of tabs that run across the top of the application.
- Job wizard infobar, which lists information about the policy, the Account, or the Line of Business.
- Job wizard toolbar, which contains Back and Next buttons to navigate through the job wizard as well as other buttons relevant to the current step.
- Screens, which contains the content for each step of the job wizard.

Reference

Job Wizard Configuration

There are four levels of configuration for job wizard

- JobWizard
- JobWizardGroup
- JobWizardStepSet and JobWizardStepRef
- JobWizardStep

JobWizard

- The **CountAsWork** property allows a user to resume working in a wizard after leaving the wizard without saving. If set to true and user navigates away before finishing a wizard, wizard step is listed as Unsaved Work within the same active user session.
- If **canEdit** evaluates to true, the job wizard is editable when in edit mode. Some conditions include: edit permission, job state.
- If **canVisit** evaluates to true, the users can view the job wizard. Some conditions include: view permission.
- Entry point definition
  - Syntax: EntryPointName(ObjectName1: ObjectType1, ObjectName2: ObjectType2 …).
  - Example: SubmissionWizard(submission : Submission, policyPeriod : PolicyPeriod).
- Each object expected in each entry points must be declared on Variables tab
- Navigating to a job wizard
  - Syntax: EntryPointOfLocation.go(objectsToPass).
  - Example: JobForward.pcf → RenewalWizard.go(job as Submission, policyPeriod, wizardStep).

A **WizardStepGroup** is a set of wizard steps which are indented and have a single parent label. For example: SubmissionWizard -> Policy Contract group.

A **WizardStepSet** is a set of wizard steps which have shared logic (visible, available, mode) and can be inserted into a wizard directly or into a wizard group.

- In the base application, there are sets of wizard steps for the submission wizard which change depending upon the line of business. For example, one set of wizard steps is used for personal auto submission and a different set of wizard steps for the workers comp submission.
- The availability and visibility logic for a wizard step set is accessible in the properties window. The step set can use the mode property to select the line of business for the submission.

A **WizardStepSetRef** is a reference to the PCF where the step set is stored.

A **JobWizardStep** can be dependent and independent. The order of the steps is determined by the order in which they appear in the job wizard PCF. Dependent steps are always in top box and Independent steps are always in bottom box. Child steps of a job wizard step group are indented.

- To make a step independent, set its independent attribute to true.
- The independent steps' title and title icon are set in the job wizard widget.
- For example, SubmissionWizard → Tools section contains the independent steps.

Screens are referenced by job wizard steps.

- Syntax: ScreenPropertyOfJobWizardStep = EntryPointOfScreen(objectsToPass).
- A screen PCF contains references to all embedded content.
- Screen related labels.
  - Title determines the text displayed in the title bar of a screen.
  - Label determines the text displayed in the sidebar.
- Buttons on screen
  - Wizard button (previous / next) behavior is handled automatically.
  - Toolbar buttons are displayed based on job type and status.

Stop

Lesson 4 Contacts and Locations

4.1 Requirements

Succeed Insurance requires that a contact with a role of Corporate Counsel can be added when required.

4.2 Configuration

Activity

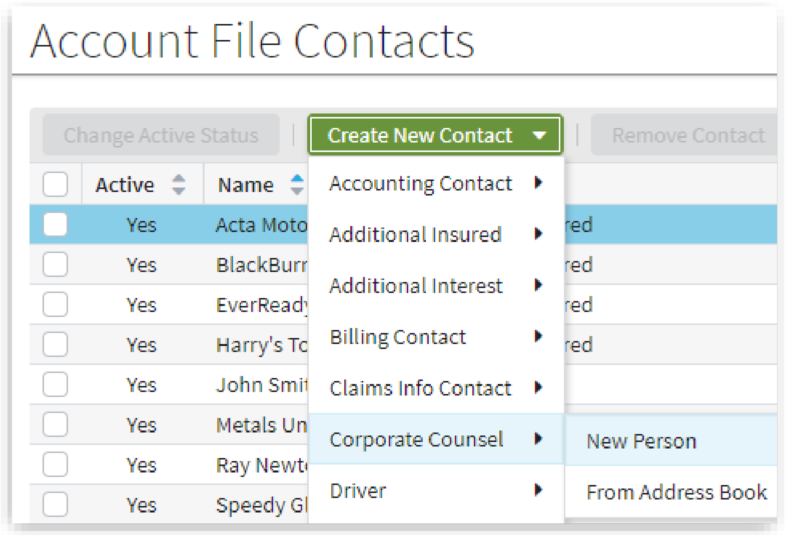
Add a new contact role Corporate Counsel.

1. **Create `CorporateCounsel_Ext` for the account contact role.**
2. **Create `PolicyCorporateCounsel_Ext` for the corresponding policy contact role.**
3. **Map policy contact role to account contact role.**
   a) Check the registered Contact Configuration Plugin in config/plugin/registry/IContactConfigPlugin.gwp.
   b) Modify the registered file, for example ContactConfigPlugin.gs or WC7ContactConfigPlugin.gs.
4. **Define the entity name and display key.**
5. **Re-start the server.**

4.3 Verification

1. **Log in to PolicyCenter as Alice Applegate, aapplegate/gw.**
2. **Navigate to Account → Wright Construction → Contacts.**

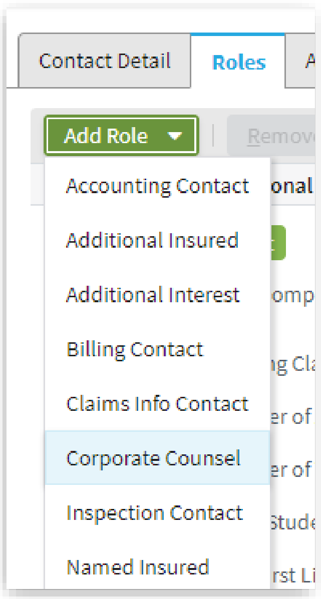
Verify that the Corporate Counsel role shows up in the Create New Contact Menu.



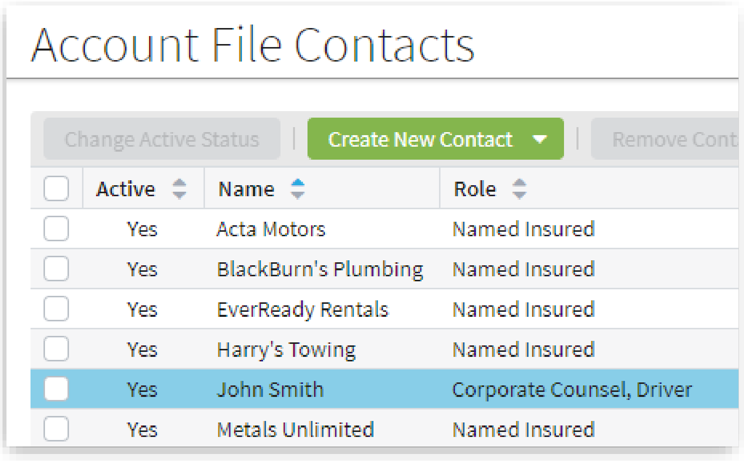
3. Add the new Corporate Counsel role to John Smith.

a) In the list view, for the contact John Smith, click the Edit icon → Roles tab → Add Role button.

b) Choose the Corporate Counsel role from the drop down and click Update to save.



4. The new role is displayed in the list view for John Smith on the Account File Contacts page.



Account File Contacts

Active	Name	Role
Yes	Acta Motors	Named Insured
Yes	BlackBurn's Plumbing	Named Insured
Yes	EverReady Rentals	Named Insured
Yes	Harry's Towing	Named Insured
Yes	John Smith	Corporate Counsel, Driver
Yes	Metals Unlimited	Named Insured

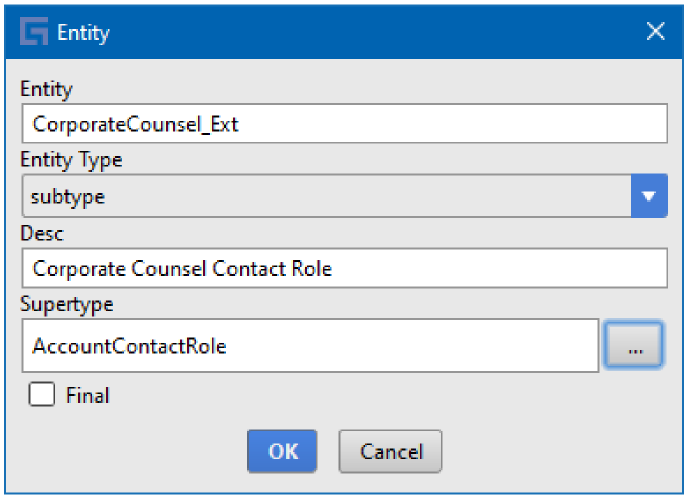
Stop

4.4 Solutions

Solution

1. **Create the account contact role.**

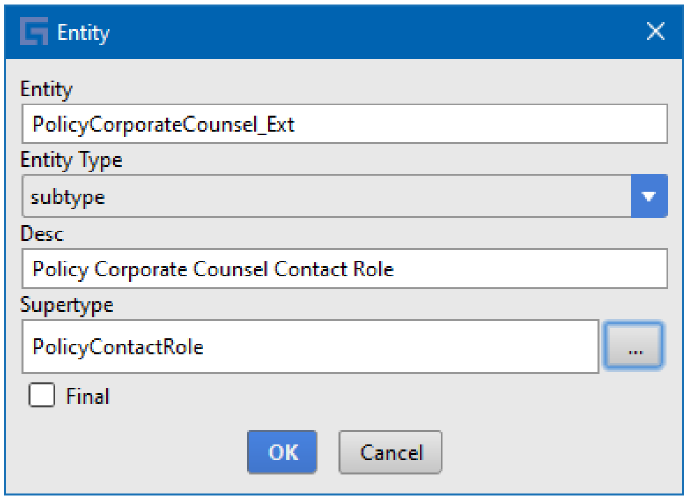
   Create `CorporateCounsel_Ext.eti` as a subtype of `AccountContactRole`.



Entity: CorporateCounsel_Ext
Entity Type: subtype
Desc: Corporate Counsel Contact Role
Supertype: AccountContactRole
Final: ☐

2. **Create the corresponding policy contact role.**

   Create `PolicyCorporateCounsel_Ext.eti` as a subtype of `PolicyContactRole`.



Entity: PolicyCorporateCounsel_Ext
Entity Type: subtype
Desc: Policy Corporate Counsel Contact Role
Supertype: PolicyContactRole
Final: ☐

3. **Map policy contact role to account contact role.**

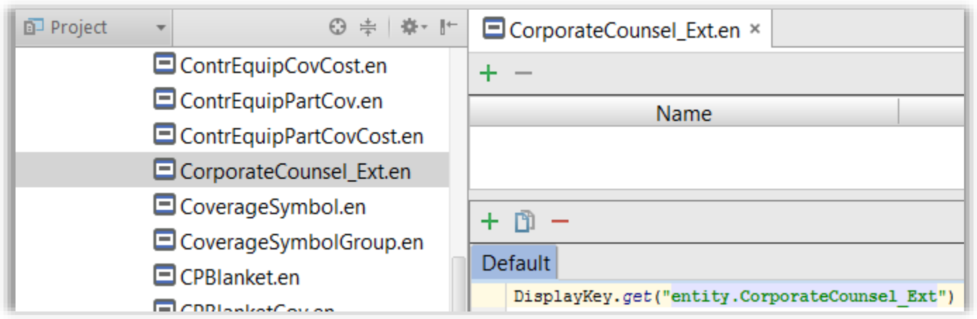
   a) Check the registered Contact Configuration Plugin in config/plugin/registry/IContactConfigPlugin.gwp.

   b) Modify the registered file, for example ContactConfigPlugin.gs or WC7ContactConfigPlugin.gs.

```
  protected property get DefaultConfigs() : ContactConfig[] {
   return {

    new ContactConfig(true, {TC_COMPANY, TC_PERSON}, TC_APDINVOLVEDPARTY,
{TC_APDPOLICYINVOLVEDPARTY}),
    new ContactConfig(true, {TC_COMPANY, TC_PERSON}, TC_CORPORATECOUNSEL_EXT,
{TC_POLICYCORPORATECOUNSEL_EXT})
    }
  }
```

4. **Define the entity name and display key.**
   a) Define the entity name for the `CorporateCounsel_Ext` entity with the correct display key.
   b) If the display key does not exist, press `Alt + Enter` to create it.



```
DisplayKey.get("entity.CorporateCounsel_Ext")
```

5. **Re-start the server.**

4.5 References

Cookbook Recipe

Steps to add a new contact role

New contact roles can be configured by following these steps:

1. **Define new subtype of `AccountContactRole`.**
2. **Define new subtype of `PolicyContactRole`.**
3. **Map policy contact role to account contact role.**
   a) Check the registered Contact Configuration Plugin in `config/plugin/registry/IContactConfigPlugin.gwp`.
   b) Modify the registed file, for example `ContactConfigPlugin.gs` or `WC7ContactConfigPlugin.gs`.

   Syntax: `new ContactConfig(enable, {contactType}, AccountContactRole,`

   `{ PolicyContactRole1, PolicyContactRole2, … } )`

   Where ContactType is the subtype of Contact (Person and Company)

4. **Define entity name for the new contact role entity**

a) Add and reference display name as needed

5. **Modify PCFs and Gosu classes as needed**
6. **Restart server to load the new entities and entity names**

Review

Contacts

Account File Contacts page lists all contacts on an account. Policy may have some or all the Account contacts. Some contact information is shared across PolicyPeriod and some is PolicyPeriod specific. Contacts are identified by the role they fill. One contact can file multiple roles on the account and policy.

Contact role behavior patterns in PolicyCenter.

- Normal: Array of contacts connected to an entity (AdditionalNamedInsured).
- Singleton: One and only one contact connected to an entity (BillingContact).
- Simple Details: One contact with array of details (AdditionalInsured).
- Join Details: Details join contact to another entity (Driver).

Review

Strategy to configure a new contact role

- For a contact role that does not exist in base application:
  - Add a new contact role and.
  - Configure it to follow the implementation of an existing contact role.
- Compare desired behavior of contact to roles in base application and find role with most similar behavior:
  - How many of them can be created?
  - How the match attaches to other roles or objects?
- Contact role entities that are extendable include:
    - PolicyContactRole and AccountContactRole.
    - All other roles in the default configuration.
    - You can define roles for Person, Company, or Both.
- Roles can be disabled when not needed.

Review

Locations

- Locations can be created on account and reused on policy and created on policy.
- **Location Numbers**: System assigns each location a sequential number stored in `LocationNum` field.
  - `AccountLocation` and `PolicyLocation` are numbered separately.

  - Configuration developers can configure the location numbering algorithm (the end user cannot change the algorithm in the UI).
- Subtypes of `PolicyLine` entity have associated location type.
  - LOB location types have foreign key Location that points to `PolicyLocation`.
- Creating a new `AccountLoction`: To add a new `AccountLocation` to the account, call the `newLocation()` method on Account and it returns the `AccountLocation` created.
- Creating a new `PolicyLocation`: To add a new `PolicyLocation`, call the `newLocation()` method on `PolicyPeriod`.
  - `newLocation()` or
  - `newLocation(acctLoc : AccountLocation)`

    where:

    `acctLoc` is the existing `AccountLocation` to which the new policy location is associated with.

Stop

Lesson 5 Concepts of Revisioning

Note: There is no lab for this module.

Stop

Lesson 6 Raising Underwriting Issues

6.1 Requirements

Configure PolicyCenter to raise an underwriting (UW) issue when a driver assigned to a vehicle has been licensed for 5 years or less.

If the driver is assigned to multiple vehicles, then an UW issue should be raised for each vehicle. The UW issue should be created before quoting and PolicyCenter should not allow the policy to be bound.

6.2 Configuration

Identify the best way to implement the configuration

- Configure a new Context Definition for `VehicleDriver` because no such iterative context is available OOTB.

  Although the Vehicle iterative context is available, and the user can access the `VehicleDriver` array through `Vehicle.Drivers`, the user cannot write loop expressions directly via `Vehicle.Drivers` in the Underwriting Rule user interface.

- Define a new an underwriting rule and the conditions to raise the UW issue.

Activity

Configure a new Underwriting Rule.

1. **Add a new context definition for each `VehicleDriver`**
   a) Add a new `RuleContextDefinitionKey` type code for the new context definition.
   b) Add the new `RuleContextDefinitionKey` type code to `UnderwriterEvalutorTriggeringPoint`
   c) Create a new Gosu class `PersonalAutoVehicelDriversIterableUWContextDefinition`, which extends `PersonalAutoUWContextDefinition`.
   d) In `PersonalAutoVehicelDriversIterableUWContextDefinition`, add a new iterable symbol for each `VehicleDriver` on each Vehicle.
   e) Add the new `RuleContextDefinitionKey` type code in `PersonalAutoVehicelDriversIterableUWContextDefinition`.
2. **Configure `BizRulesPlugin`**

   Add the new context definition to `BizRulesPlugin`.
3. **Calculate `YearsLicensed`**

   Add a new property in `PolicyDriverEnhancement`.

4. **Add a new underwriting rule**
    a) Go to Administration -> Business Settings -> Business Rules -> Underwriting Rules.
    b) Click **Add**.
5. **Specify the values to define a new UW rule**

    In the Rule **Details** card view, fill out the required fields, especially checking set, blocking point to meet the requirements. In the **Advanced** Card, select *At Least* as the value comparator and set the other fields.

Important!

The Code field must be unique among all the underwriting rules.

1. **Define the rule context and conditions to raise the UW issue**
    a) In the **Applies to** section, select Personal Auto Line. If required, select the appropriate Jurisdiction and Policy Transactions.
    b) In the **Rule Context** section, select Personal Auto Policy – For Each Vehicle Driver.
    c) In the **Rule Condition** section, enter the expressions to check for licensed years of each driver on each vehicle.
2. **Define UW issue details including Issue Key, Short and Long Description**
    a) Enter a unique Issue Key.
    b) Enter the Short and Long Description.
3. **Click Save**

6.3 Verification

1. **Log in to PolicyCenter as Alice Applegate.**
2. **Create a 6 month Personal Auto submission for Ray Newton.**
    - Create 3 drivers with no accidents or violations:
    - One driver should be licensed for only 1 year,
    - Second driver between 2 and 5 years, and
    - Third driver is licensed for more than 5 years.
    - Create 2 vehicles and assign all three drivers to each vehicle.
    - Select standard coverages for each vehicle on this policy.
3. **Verify the behavior by clicking quote.**

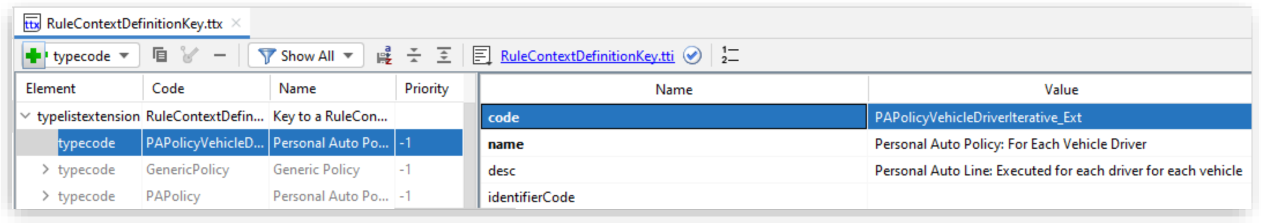
Stop

6.4 Solutions

Solution

1. **Extend the `RuleContextDefinitionKey` typelist and add a new typecode**

```
code: PAPolicyVehicleDriverIterative_Ext
name: Personal Auto Policy: For Each Vehicle Driver
desc: Personal Auto Line: Executed for each driver for each vehicle
```



RuleContextDefinitionKey.ttx

typecode | Show All | RuleContextDefinitionKey.tti

Element	Code	Name	Priority
typelistextension	RuleContextDefin...	Key to a RuleCon...	
typecode	PAPolicyVehicleD...	Personal Auto Po...	-1
typecode	GenericPolicy	Generic Policy	-1
typecode	PAPolicy	Personal Auto Po...	-1

Name	Value
code	PAPolicyVehicleDriverIterative_Ext
name	Personal Auto Policy: For Each Vehicle Driver
desc	Personal Auto Line: Executed for each driver for each vehicle
identifierCode	

2. **Add the new `RuleContextDefinitionKey typecode` to `UnderwriterEvaluatorTriggeringPoint.gs`.**

```
class UnderwriterEvaluatorTriggeringPoint implements ITriggeringPoint {


 override function supportedContexts(): Set<RuleContextDefinitionKey> {
  return {
    TC_GENERICPOLICY,
    TC_PAPOLICY,
    TC_PAPOLICYDRIVERITERATIVE,
    TC_PAPOLICYVEHICLEITERATIVE,
    TC_PAPOLICYVEHICLEDRIVERITERATIVE_EXT,

  ......
  }
 }
 ......
}
```

3. **Add a new context definition for the `VehicleDriver` Array.**

```
package gw.bizrules.provisioning.contexts

@Export
class PersonalAutoVehicleDriversIterableUWContextDefinition
   extends PersonalAutoUWContextDefinition {
  public static final var PARAM_VEHICLE_DRIVER: String = "paVehicleDriver"

  construct() {
   addIterativeSymbol(PARAM_VEHICLE_DRIVER, VehicleDriver,
              \ec -> ec.Period.PersonalAutoLine.Vehicles*.Drivers)
  }

  override property get Key(): RuleContextDefinitionKey {
   return RuleContextDefinitionKey.TC_PAPOLICYVEHICLEDRIVERITERATIVE_EXT
  }
```

```
      }
```

4. **Configure `BizRulesPlugin` to add the new context definition.**

```
uses gw.bizrules.provisioning.contexts.PersonalAutoVehicleDriversIterableUWContextDefinition

class BizRulesPlugin implements IBizRulesPlugin {
 private final var _uwContexts: IRuleContextDefinition[]

 construct() {
  _uwContexts = {
    new GenericUWRuleContextDefinition(),
    new PersonalAutoUWContextDefinition(),
    new PersonalAutoDriversIterableUWContextDefinition(),
    new PersonalAutoVehiclesIterableUWContextDefinition(),
    new PersonalAutoVehicleDriversIterableUWContextDefinition(),

     ……
   }
  }
 }
```

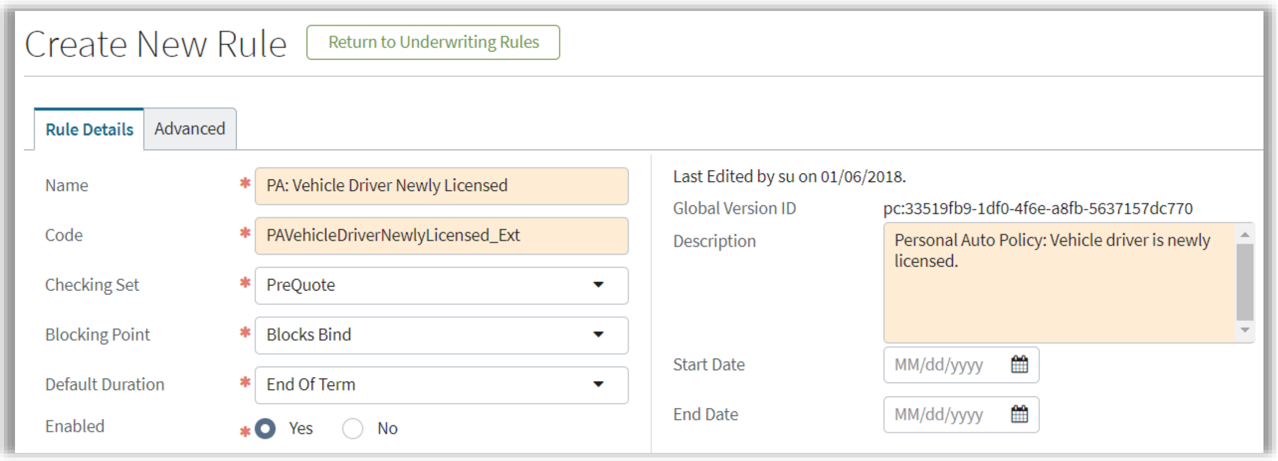
5. **Add a new property to `PolicyDriverEnhancement` to get the number of years licensed.**

```
enhancement PolicyDriverEnhancement : entity.PolicyDriver {

  property get YearsLicensed_Ext(): int {
    var yearLicensed = (this.AccountContactRole as Driver).YearLicensed
    var yearsLicensed = yearLicensed == null ? 0:
           (Date.CurrentDate.YearOfDate - yearLicensed + 1)
    return yearsLicensed
  }

}
```

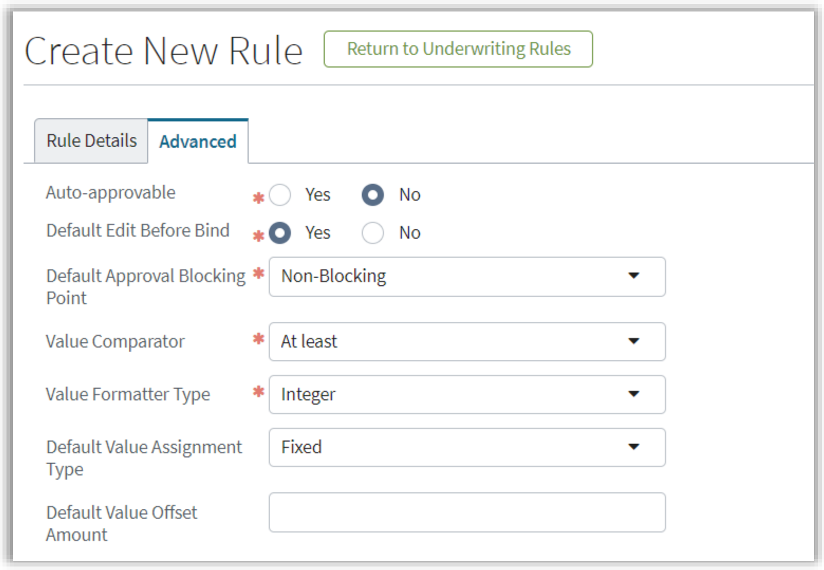
6. **Restart the server.**
7. **Log in as super user and go to Administration → Business Settings -> Underwriting Rules.**
8. **Add a new Underwriting Rule with basic and advanced values.**



Name *	PA: Vehicle Driver Newly Licensed
Code *	PAVehicleDriverNewlyLicensed_Ext
Checking Set *	PreQuote
Blocking Point *	Blocks Bind
Default Duration *	End Of Term
Enabled *	[✓ Yes] No

Last Edited by su on 01/06/2018.

Global Version ID	pc:33519fb9-1df0-4f6e-a8fb-5637157dc770
Description	Personal Auto Policy: Vehicle driver is newly licensed.
Start Date	MM/dd/yyyy
End Date	MM/dd/yyyy

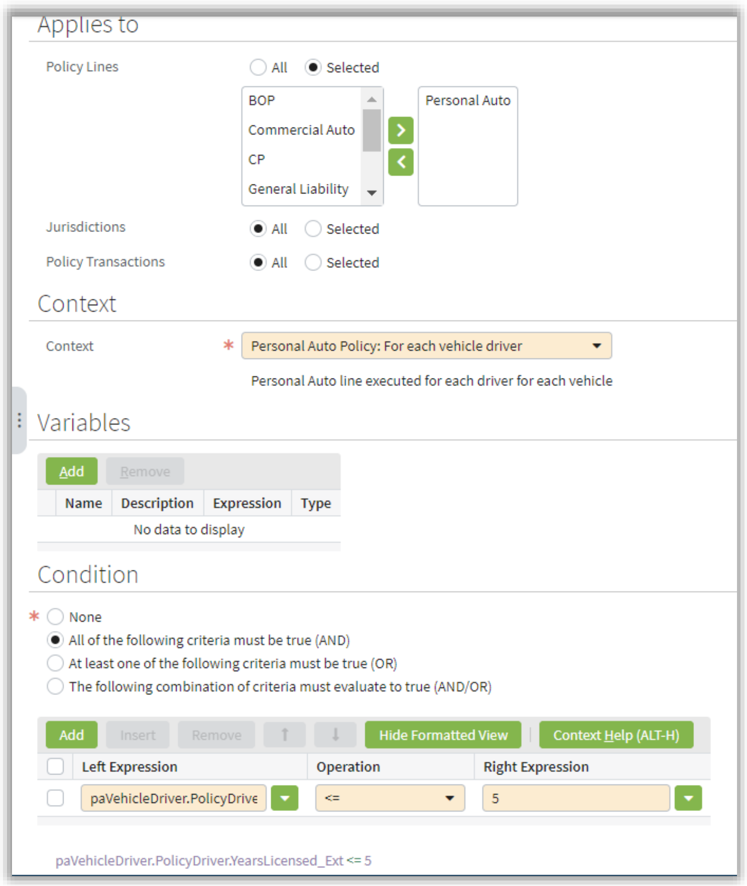

Create New Rule Return to Underwriting Rules

Rule Details | Advanced

Auto-approvable *	Yes [✓ No]
Default Edit Before Bind *	[✓ Yes] No
Default Approval Blocking Point *	Non-Blocking
Value Comparator *	At least
Value Formatter Type *	Integer
Default Value Assignment Type	Fixed
Default Value Offset Amount	

9. **Select the policy lines, rule context, and define the conditions to raise the UW issue on the Rule Details tab.**

```
Left Expression: paVehicleDriver.PolicyDriver.YearsLicensed_Ext
```


Applies to

Policy Lines: All [✓ Selected]

Available: BOP, Commercial Auto, CP, General Liability — Selected: Personal Auto

Jurisdictions: [✓ All] Selected

Policy Transactions: [✓ All] Selected

Context

Context *: Personal Auto Policy: For each vehicle driver

Personal Auto line executed for each driver for each vehicle

Variables

Add Remove

Name	Description	Expression	Type
No data to display			

Condition

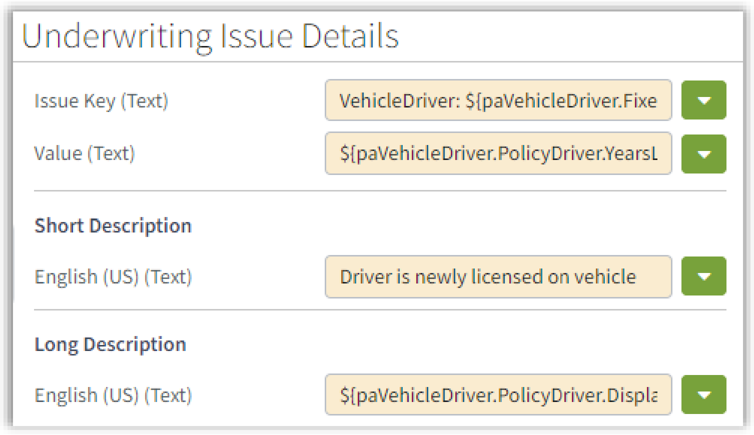
* None
[✓ All of the following criteria must be true (AND)]
At least one of the following criteria must be true (OR)
The following combination of criteria must evaluate to true (AND/OR)

Add Insert Remove Hide Formatted View Context Help (ALT-H)

Left Expression	Operation	Right Expression
paVehicleDriver.PolicyDrive	<=	5

paVehicleDriver.PolicyDriver.YearsLicensed_Ext <= 5

10. Define the UW issue details.



Underwriting Issue Details

Issue Key (Text)	VehicleDriver: ${paVehicleDriver.Fixe
Value (Text)	${paVehicleDriver.PolicyDriver.YearsL
Short Description	
English (US) (Text)	Driver is newly licensed on vehicle
Long Description	
English (US) (Text)	${paVehicleDriver.PolicyDriver.Displa

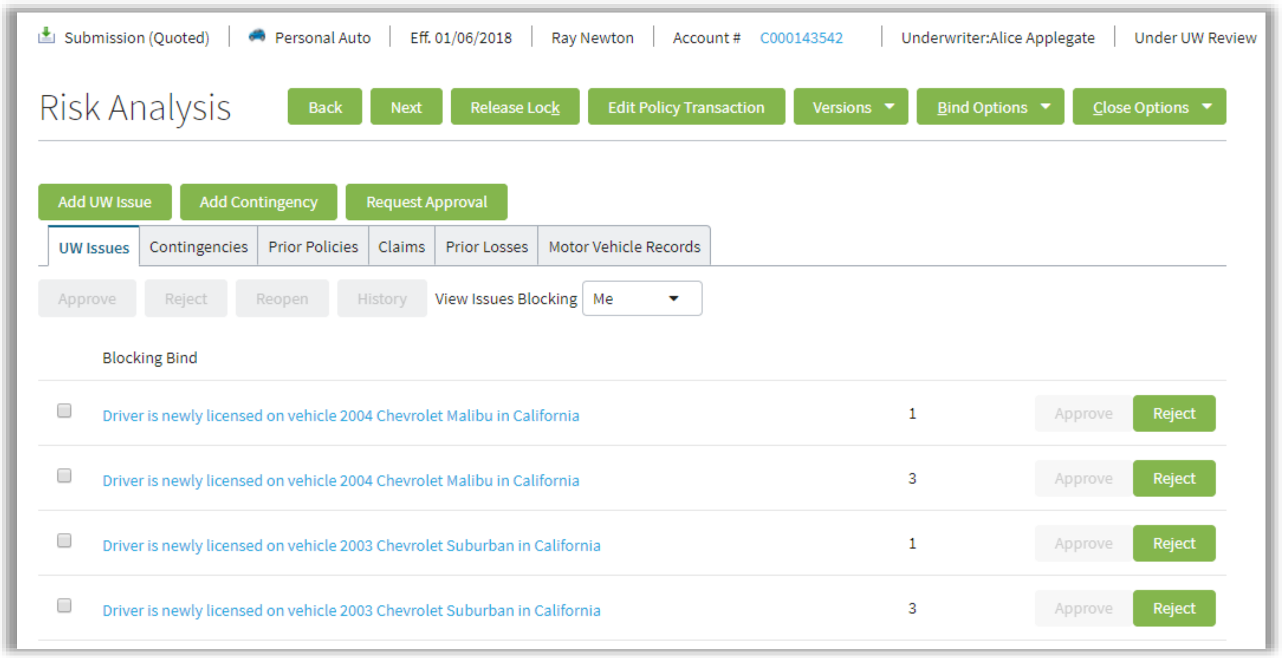
Code for your reference.

- IssueKey: `VehicleDriver: ${paVehicleDriver.FixedId}`
- Value: `${paVehicleDriver.PolicyDriver.YearsLicensed_Ext}`
- Short Description: `Driver is newly licensed on vehicle ${paVehicleDriver.Vehicle.DisplayName}`
- Long Description: `${paVehicleDriver.PolicyDriver.DisplayName} is newly licensed for ${paVehicleDriver.PolicyDriver.YearsLicensed_Ext} year(s).`
- `Fill in all language fields.`

11. Log out and log in to PolicyCenter as Alice Applegate.
12. Create a 6 month Personal Auto submission for Ray Newton.

- Create 3 drivers with no accidents or violations:
  - One driver should be licensed for only 1 year,
  - Second driver between 2 and 5 years, and
  - Third driver is licensed for more than 5 years.
- Create 2 vehicles and assign all three drivers to each vehicle.
- Select standard coverages for each vehicle on this policy.

13. Click quote and verify that the underwriting issue is raised.



Blocking Bind			
Driver is newly licensed on vehicle 2004 Chevrolet Malibu in California	1	Approve	Reject
Driver is newly licensed on vehicle 2004 Chevrolet Malibu in California	3	Approve	Reject
Driver is newly licensed on vehicle 2003 Chevrolet Suburban in California	1	Approve	Reject
Driver is newly licensed on vehicle 2003 Chevrolet Suburban in California	3	Approve	Reject

6.5 References

6.5.1 Configuring a new underwriting rule

Cookbook Recipe

Steps to configure a new underwriting rule

1. **Configure Business Rule Context Definition**

   a) To add new symbols to existing Context Definition:

   - Open an existing `UWRuleContextDefinition` Gosu class in Guidewire studio.
   - In the constructor, add a new line of code to call `addSymbol()` and pass the symbol name, the object type and the block code to extract the actual instance of the object for the symbol.

   b) To add a new Context Definition class:

   - Extend one of the existing parent Context Definition class (e.g. `PAAutoUWContextDefinition`) or create a new class that implements the `UWRuleContextDefinition` interface.
   - Implement the necessary functions to specify the LOB it applies to and the symbols to be added.
   - Add a new type code in the `RuleContextDefinitionKey` typelist if necessary.
   - Add the new `RuleContextDefinitionKey` type code to `UnderwriterEvaluatorTriggerPoint`.

2. **Configure `BizRulesPlugin`**

   a) Add the new Context Definition class to `BizRulesPlugin`.

   b) Configure the blacklisted or whitelisted methods to hide or expose properties and methods of the contexts.

c) Some methods may be added to the `PolicyContextDefinitionLibrary` or the related entity first to expose values that are not supported in the UI, for example object casting.

3. Add a new underwriting rule

a) Go to Administration → Business Settings → Business Rules → Underwriting Rules.

b) Click Add.

4. Specify basic values defining a new UW issue rule

a) In the top section of the Rule Details card, fill out the fields related to UW issue type.

5. Specify the advanced values of the new UW issue

a) In the Advanced card view, fill out the fields used for Value Comparator and approval, such as Auto-approvable, Default Edit Before Bind.

6. Define the rule context and conditions to raise the UW issue

a) In the applicability section, select the Policy Lines, Jurisdictions, and Policy Transactions that this rule applies to.

b) In the Rule Context section, set the context that this rule will be run with.

c) In the Rule Condition section, create the conditions used to raise the UW issue. Use the context selected above to create expressions. Multiple rows of criteria can be combined using the AND/OR operands.

7. Define UW issue details including Issue Key, Short and Long Description

a) In the UW Issue Details section, create an Issue Key that can uniquely identify the UW Issue once it is raised. For example, a fixedID of any Gosu object or the social security number of a contact object.

b) Enter a short and a long description for the UW Issue.

Tip

- What Gosu Expressions are not supported in the Underwriting Rule user interface?

  These Gosu Expressions are not supported in the Underwriting Rule user interface: casting, loops, assignment, block code, and array element (e.g. drivers[0]).

- Methods of any available context symbols can be accessed in the UI. If any method is hidden (e.g. object casting), it can be exposed by adding it to the WhiteListedMethods in BizRulesPlugin or directly in the enhancement class of the entity.

Find

For more information about the UW Rule fields, refer to the PolicyCenter Application Guide.

6.5.2 Create underwriting issues in Gosu code

Reference

Underwriting issues can be raised by authoring the rule condition in the Gosu code instead of in the Underwriting Rules user interface.

Note: Refer to the PolicyCenter Configuration Guide for more information on how to create underwriting issues in Gosu code.

Pros and Cons of the two approaches:

Underwriting rules conditions written using the Underwriting Rule screen can be viewed by authorized users in PolicyCenter. These rule conditions can be edited, enabled, and disabled, and can be efficiently moved between PolicyCenter instances. Underwriting rule conditions written in Gosu are not accessible in the user interface. However, Gosu code can handle more complex rules and, in some cases, improve system performance. In both cases, PolicyCenter creates UWRule and UWIssueType entities. These entities have foreign keys to each other.

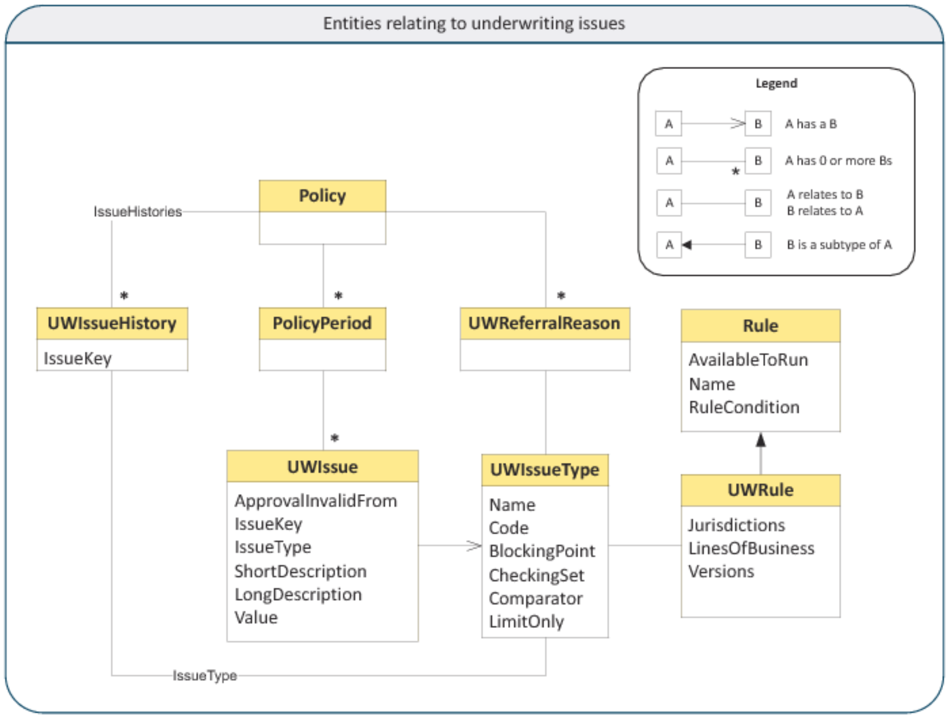
Important!

To avoid unexpected results, specify the rule condition in either the user interface or the Gosu code, but not in both. PolicyCenter first executes the Rule Condition in the user interface and raises an underwriting issue when the condition evaluates to true. Then PolicyCenter executes the rule condition in the Gosu. If the Rule Condition in the user interface created an underwriting issue, the rule condition in the Gosu code does not raise another issue.

A Gosu rule still needs a corresponding UWRule so that it creates an UWIssueType to be referenced in the Gosu code. In this case, the UWRule is usually disabled or has None selected in the Rule Condition in the user interface.

6.5.3 UWIssue object model

Reference



The diagram shows some entities associated with underwriting issues:

- PolicyPeriod - Stores information for a specific period of a policy.
- UWIssue - These are the issues that evaluator classes create. The issues link to issue types and can contain an approval. These entities can vary in effective time and are copied from branch-to-branch, just like any other effective-dated entity. Some fields on this entity are:
  - IssueKey and IssueType – These two fields uniquely identify the issue.
  - ShortDescription – Description of the issue that appears in the user interface.
  - LongDescription – Long description of the issue that appears in the user interface. Some users may have permission to view the short description but not the long description.
  - Value – The value, if any, associated with this issue. If present, the value is compared with authority grants to determine if the user can approve this issue. The value is also compared with approvals to determine if the approval still applies.
  - Approval - This is an UWIssueApproval object and represents the approval or rejection of an issue. Approvals can be generated either by a user clicking to approve or reject an issue, or automatically by the application. At any point in effective time, each UWIssue can have at most one approval.
    - **Rejections**: Rejections generate an approval, in case it does not exist already, but with the `BlockingPoint` set to `Rejected`. The user can reject an issue if:
      - It is eligible for approval.
      - It is a non-blocking issue and has not already been rejected.
    - **Reopening**: Reopening an issue removes an approval from the issue. The issue then blocks at the blocking point associated with the issue type.
    - **Special Approve**: Available to users such as `su` with the `uwapproveall` permission but do not have the authority to approve the issue.
  - UWIssueHistory - This entity represents a history of issues and approvals on the policy. The IssueKey field is a reference to the UWIssue with that IssueKey. PolicyCenter maintains the history in a set of non-

effdated entities on the policy and outside of the context of any particular job or policy period. PolicyCenter maintains this history for the following reasons:

- The entity captures events that happen on branches that do not end up binding.
- The entity persists after issues are removed or approvals expire.
- The entity persists even if the related periods are archived.

Since this entity attaches directly to the policy, it is retireable rather than effdated.

- UWIssueType - This entity defines issue types.
- UWRule – This entity has foreign key to UWIssueType and contains the Jurisdiction and Line Of Business.
- Rule – The super type of UW Rule. Contains the Rule Condition used to check and raise UW issues.
- UWReferralReason - This entity is similar as the UWIssue entity but exists off of the policy rather than the policy period. This entity is used for marking a policy issue outside of a job. It identifies the details of a corresponding UWIssue that will be generated the next time the application checks for UWIssues. Because of the close relationship between an UWReferralReason and the associated UWIssue, the default user interface displays them in a similar way.

Stop

Lesson 7 Approving Underwriting Issues

7.1 Requirements

The underwriting issue that is created in the Raising Underwriting Issue lesson should be approved by a user with an Authority Profile of Underwriter 1. The authority grant should allow the underwriter to approve a driver licensed for two or more years. The approval should be valid for the remainder of the policy term and the approval should allow for the submission to be edited after approval.

7.2 Configuration

Activity

Configure PolicyCentert to approve an UW Issue

1. **Specify the values related to approve the UW issue**

   Update the new UW rule for newly licensed driver. In the Basic and Advanced card view, fill out the required fields to meet the requirements for default duration, comparators, and other approval criteria if you did not set it during the previous lab.

2. **Grant Authority to Approve the new UW issue**
   a) Log in to PolicyCenter as su/gw.
   b) Edit the Authority Profile `Underwriter 1`.
   c) Add a new Authority Grant to approve the new UW issue for a driver who has been licensed for at least 2 years.
   d) Save `Underwriter 1`.

3. **Assign the Authority Profile to user**
   a) User Alice Applegate should already have the Authority Profile `Underwriter 1` assigned to her.
   b) If not, assign `Underwriter 1` to user Alice Applegate.

7.3 Verification

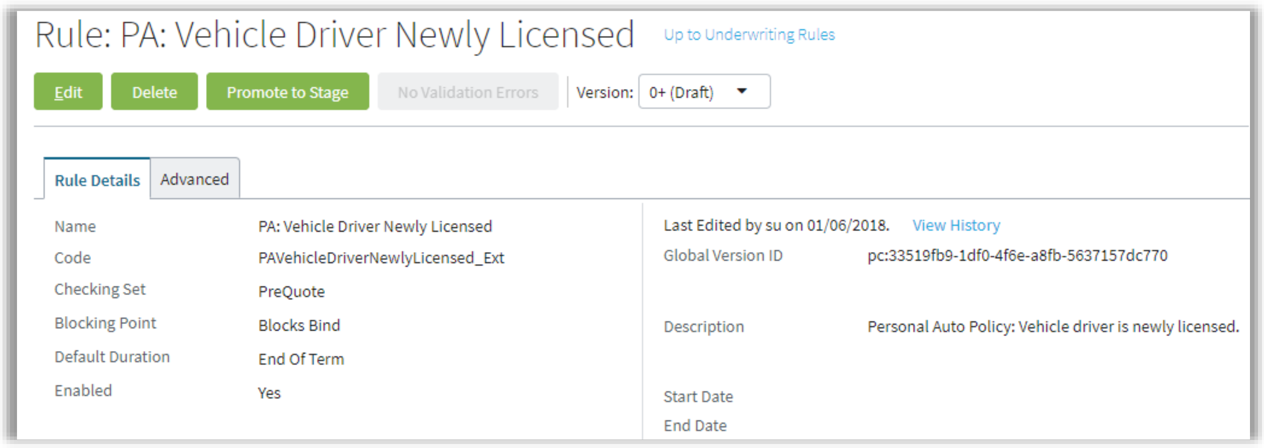
1. **Log in to PolicyCenter as Alice Applegate.**
2. **Go to the previous Personal Auto submission for Ray Newton that raised the UW issue for newly licensed driver.**
3. **Verify that Alice can approve the UW issue on the Risk Analysis page.**

Stop

7.4 Solutions

Solution

1. **Log in as super user.**
2. **Update Underwriting Rule.**
   a) Go to Administrations -> Business Rules -> Underwriting Rules.
   b) Find the underwriting rule and update it with basic and advanced values to meet the requirement.

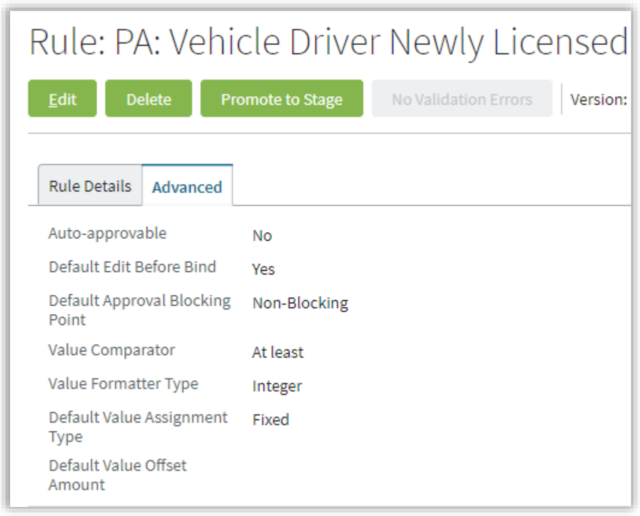


Version: 0+ (Draft)

Name	PA: Vehicle Driver Newly Licensed
Code	PAVehicleDriverNewlyLicensed_Ext
Checking Set	PreQuote
Blocking Point	Blocks Bind
Default Duration	End Of Term
Enabled	Yes

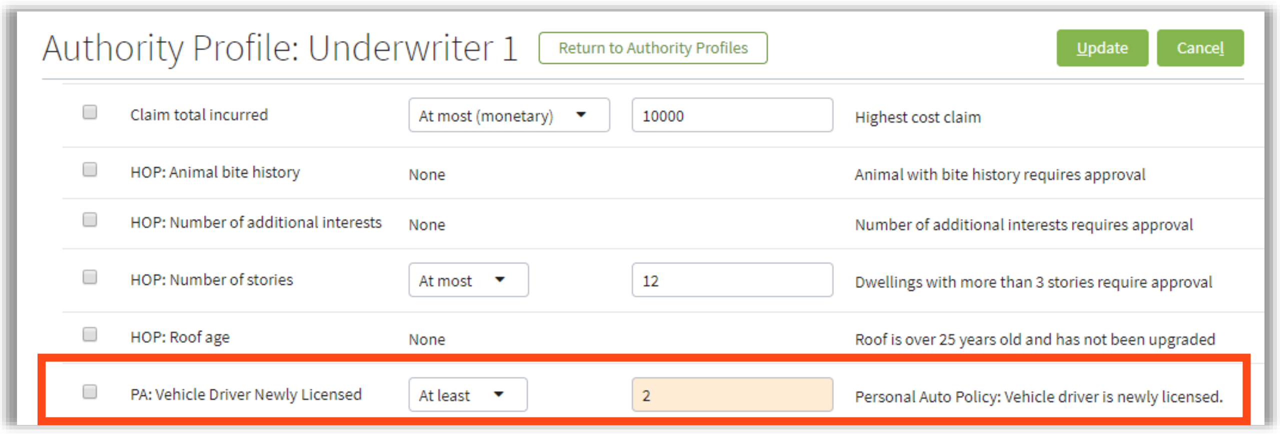
Last Edited by su on 01/06/2018.

Global Version ID	pc:33519fb9-1df0-4f6e-a8fb-5637157dc770
Description	Personal Auto Policy: Vehicle driver is newly licensed.
Start Date	
End Date	



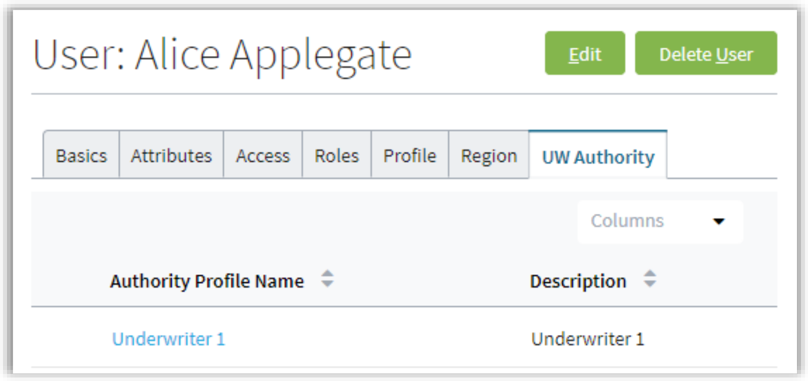
Auto-approvable	No
Default Edit Before Bind	Yes
Default Approval Blocking Point	Non-Blocking
Value Comparator	At least
Value Formatter Type	Integer
Default Value Assignment Type	Fixed
Default Value Offset Amount	

3. **Manage Authority Profile**
   a) Go to Users and Security -> Authority Profiles.
   b) Edit **Underwriter 1** and grant authority to approve the new issue.


Authority Profile: Underwriter 1

☐	Claim total incurred	At most (monetary)	10000	Highest cost claim
☐	HOP: Animal bite history	None		Animal with bite history requires approval
☐	HOP: Number of additional interests	None		Number of additional interests requires approval
☐	HOP: Number of stories	At most	12	Dwellings with more than 3 stories require approval
☐	HOP: Roof age	None		Roof is over 25 years old and has not been upgraded
☐	PA: Vehicle Driver Newly Licensed	At least	2	Personal Auto Policy: Vehicle driver is newly licensed.

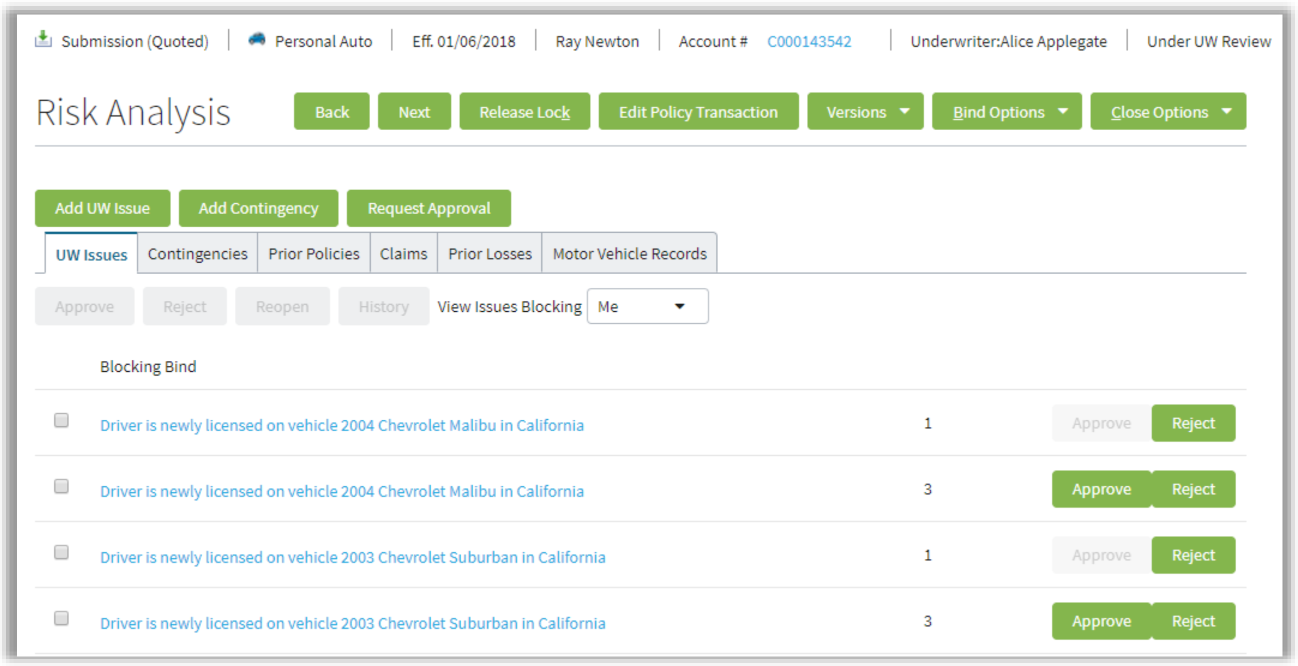
4. **Assign `Underwriter 1` to the users if not done yet.**


User: Alice Applegate

Authority Profile Name	Description
Underwriter 1	Underwriter 1

5. **Log out and log in as Alice Applegate.**
6. **Go to the previous Personal Auto submission and approve the UW Issue.**

   Since aapplegate has the authority profile of Underwriter 1, she can approve the issue when the driver is licensed for 3 years (Approve button is available) but cannot approve when a driver is licensed for one year only.

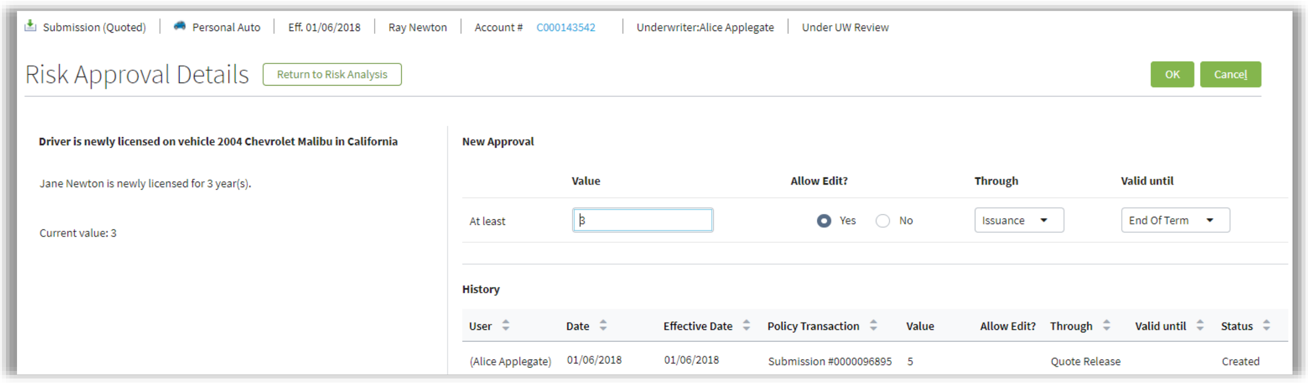


Submission (Quoted) | Personal Auto | Eff. 01/06/2018 | Ray Newton | Account # C000143542 | Underwriter:Alice Applegate | Under UW Review

Risk Analysis

View Issues Blocking: Me

Blocking Bind		
Driver is newly licensed on vehicle 2004 Chevrolet Malibu in California	1	Approve Reject
Driver is newly licensed on vehicle 2004 Chevrolet Malibu in California	3	Approve Reject
Driver is newly licensed on vehicle 2003 Chevrolet Suburban in California	1	Approve Reject
Driver is newly licensed on vehicle 2003 Chevrolet Suburban in California	3	Approve Reject



Submission (Quoted) | Personal Auto | Eff. 01/06/2018 | Ray Newton | Account # C000143542 | Underwriter:Alice Applegate | Under UW Review

Risk Approval Details

Driver is newly licensed on vehicle 2004 Chevrolet Malibu in California

Jane Newton is newly licensed for 3 year(s).

Current value: 3

New Approval

	Value	Allow Edit?	Through	Valid until
At least	3	[✓ Yes] No	Issuance	End Of Term

History

User	Date	Effective Date	Policy Transaction	Value	Allow Edit?	Through	Valid until	Status
(Alice Applegate)	01/06/2018	01/06/2018	Submission #0000096895	5		Quote Release		Created

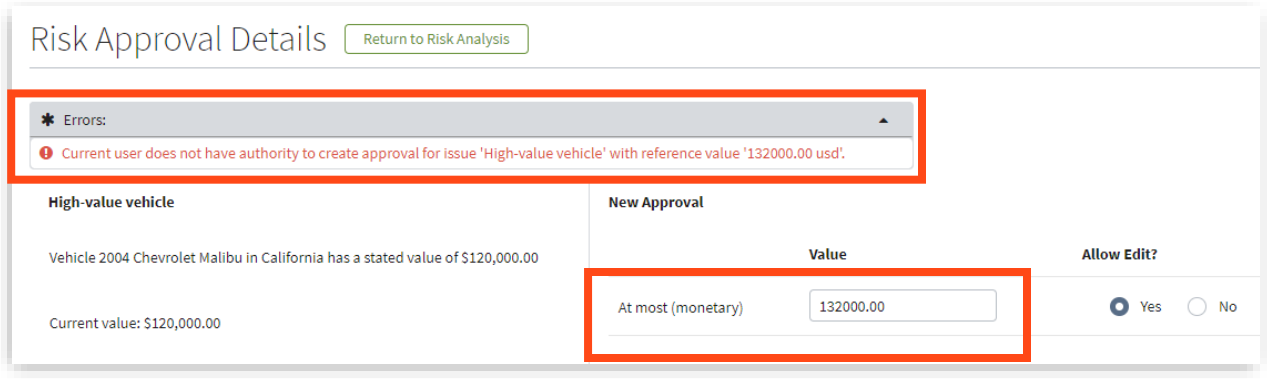
7.5 References

7.5.1 Authority Profile limits what a user can approve

For example, a vehicle is valued at $120,000. Let's say the authority profile is set as At Most $125,000 and in the UW rule for High-value Vehicle, we set the offset approval to 10%.

The approval screen shows an At Most value of $132,000 (120,000 + 10% of 120,000 = 120,000 + 12,000 = 132,000) defined in the Underwriting Rule. However, since the UW only has an approval limit of $125,000, they will need to change that $132,000 value to $125,000 before they can approve otherwise the system will throw an error as shown in the screenshot.

- Try to approve the issue and an error occurs because the calculated value 132,000 is greater than the limit 125, 000 that the underwriter's authority profile has.



Risk Approval Details

Return to Risk Analysis

Errors:

Current user does not have authority to create approval for issue 'High-value vehicle' with reference value '132000.00 usd'.

High-value vehicle

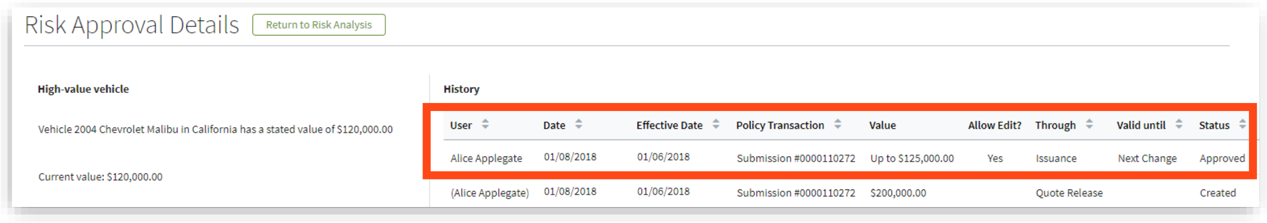
Vehicle 2004 Chevrolet Malibu in California has a stated value of $120,000.00

Current value: $120,000.00

New Approval

	Value	Allow Edit?
At most (monetary)	132000.00	[✓ Yes] No

- Change value on the approval screen from 132,000 to 125,000, then UW can approve the issue.



Risk Approval Details

Return to Risk Analysis

High-value vehicle

Vehicle 2004 Chevrolet Malibu in California has a stated value of $120,000.00

Current value: $120,000.00

History

User	Date	Effective Date	Policy Transaction	Value	Allow Edit?	Through	Valid until	Status
Alice Applegate	01/08/2018	01/06/2018	Submission #0000110272	Up to $125,000.00	Yes	Issuance	Next Change	Approved
(Alice Applegate)	01/08/2018	01/06/2018	Submission #0000110272	$200,000.00		Quote Release		Created

Stop

Lesson 8 Validation Classes

8.1 Validation chaining

8.1.1 Investigation

Observe the validation sequence and validation chaining used in the business auto line of business.

Activity

Answer the following questions by drilling into the `BALineValidation.gs` class for the Commercial Auto line in Studio.

a) List the primary validate method in that class.

b) List all methods called from the validate method in sequence.

Stop

8.1.2 Solutions

Solution

Answer the following questions by drilling into the `BALineValidation.gs` class for the Commercial Auto line in Studio.

a) List the first method that is called in that class?

`doValidate()`

b) List all methods called from the validate method in sequence:

- *`additionalInsuredAndTypeUnique()`*
- *`vinIsUnique()`*
- *`fleetTypeMatchesVehicleInfo()`*
- *`checkLiabilityCoverage()`*
- *`atLeastOneHiredAutoState()`*
- *`atLeastOneNonownedState()`*
- *`nonOwnedBasisSumGreaterThanZero()`*
- *`baLine.Vehicles.each( \ vehicle -> new BusinessVehicleValidation(Context, vehicle).validate() )`*

8.2 Configure validation classes

8.2.1 Requirement

Succeed Insurance wants to make sure that users cannot buy Personal Auto insurance for more than two vehicles.

Specifications:

Spec 1 Add a validation check that warns users when they create more than two vehicles that they are not allowed to do that.

Spec 2 The warning must appear when the user leaves the Vehicles Wizard step.

Spec 3 The validation must prevent users from quoting the policy when there are more than two vehicles on the policy.

8.2.2 Configuration

Activity

The following are some tips:

1. **PALineValidation uses a class called PALineVehiclesValidator.**

   The method can be placed there.

2. **Validation level typecodes should be used instead of the level code as a string, for example:**

   ```
   default → ValidationLevel.TC_DEFAULT
   quickquote → ValidationLevel.TC_QUICKQUOTABLE
   ```

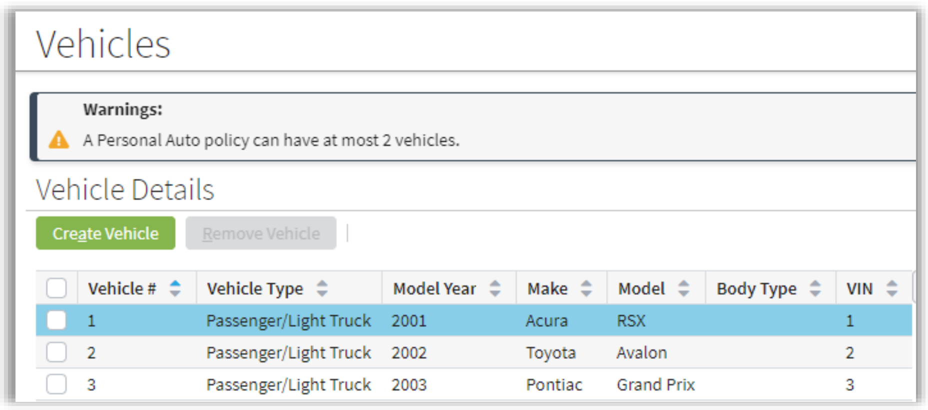
   and so on…

8.2.3 Verification

1. **Log in as aapplegate/gw.**
2. **Create a personal auto submission with three vehicles.**

   You can open an existing policy and click Actions → Copy Submission, and then edit it to have three vehicles. Or create a new submission with three vehicles.

3. **When you leave the Vehicles step of the job wizard, the following warning should be displayed.**



Vehicles

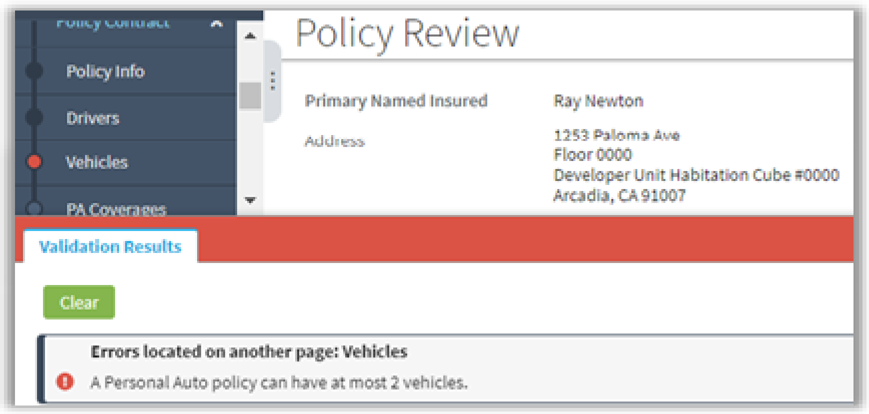
Warnings:
A Personal Auto policy can have at most 2 vehicles.

Vehicle Details

Vehicle #	Vehicle Type	Model Year	Make	Model	Body Type	VIN
1	Passenger/Light Truck	2001	Acura	RSX		1
2	Passenger/Light Truck	2002	Toyota	Avalon		2
3	Passenger/Light Truck	2003	Pontiac	Grand Prix		3

4. **Move through the remaining wizard steps until you can quote the policy.**

   When you click the Quote button, your validation class should display an error with a link to the Vehicles page as shown in the following screen. A quote should not be displayed. The error should prevent you from getting a quote.



Policy Review

Primary Named Insured: Ray Newton

Address: 1253 Paloma Ave
Floor 0000
Developer Unit Habitation Cube #0000
Arcadia, CA 91007

Errors located on another page: Vehicles
A Personal Auto policy can have at most 2 vehicles.

Stop

8.2.4 Solutions

Solution

1. **In `PALineVehiclesValidator.gs`, add a function to check for at most two vehicles and add the method to the function `doValidate()`.**

```
class PALineVehiclesValidator extends PolicyLineValidation<entity.PersonalAutoLine> {

  override function doValidate() {
   atLeastOneVehicle()
   atMostNumVehicles(2)


  }

  public function atMostNumVehicles(num: int) {
   Context.addToVisited(this, "atMostTwoVehicles()")
   if (paLine.Vehicles.Count > num and Context.isAtLeast(ValidationLevel.TC_DEFAULT)){
    var msg = DisplayKey.get("Ext.Validator.AtMostTwoVehicles", num)
    if ( Context.isAtLeast(ValidationLevel.TC_QUICKQUOTABLE))
     Result.addError(paLine, ValidationLevel.TC_QUICKQUOTABLE, msg, VEHICLES_WIZARD_STEP)
    else
     Result.addWarning(paLine, ValidationLevel.TC_DEFAULT, msg, VEHICLES_WIZARD_STEP)
   }
  }

}
```

2. **Create the display key.**

   Press Alt-Enter on the display key used in the method above and create the new display key.

   `Ext.Validator.AtMostTwoVehicles = A Personal Auto policy can have at most {0} vehicles.`

3. **Stop and restart server using Run -> Debug Server or simply Reload Changed Classes in Studio.**

8.3 References

Review

Creating a new validation class:

New class should extend `PCValidationBase` or `PCValidation`. Use an existing validation class as a reference.

For new entities:

- Create new validation class for entity you want to validate or
- Create new validation class when you create a new LOB

For existing entities:

- If a validation class for the entity exists add validation check methods to that class
- If the validation class does not exist, then create new validation class

Review

Steps to add a validation check:

1. **Edit/add appropriate validation class.**

   First you will have to decide whether this validation can be added to an existing class or a new validation class needs to be created. If, on the other hand, you needed to add a validation check for a new entity that didn't already have any validation checks and no validation checks exist for it, you would need to create a new validation class to validate that specific entity.

2. **Add the validation check method.**

   You will need to restart the server because of display key addition.

3. **Call method from `validationImpl()`.**

   The `validateImpl` method of any class calls all the methods in a logical sequence.

   Note: In case where you had to create a new validation class then you will also have to create a `validateImpl()` method and then add the method call statement to the method. For the example, we are creating a validation check for a personal vehicle and adding the method call statement to the `validateImpl()` method of the `PersonalVehicleValidation` class.

   If you are extending `PolicyLineValidation` class, you will have to create a `doValidate()` method and call the methods that validate a single issue in a logical order.

4. **Invoke the method.**

   The validation class methods must be invoked explicitly and are not called automatically. If it is already in the validation chaining path, you do not have to invoke it.

5. **Verify the result in the UI.**

Review

Invoking class-based validation:

Class-based validation must be explicitly invoked through code, job processes, wizard steps, workflow steps, integration plug-ins, or from any Gosu code.

To invoke validation from a particular PolicyCenter location:

- Wizard steps - use beforeSave attribute for that step.
- Pop-ups - use beforeCommit attribute for that pop-up.

Review

Validation Chaining is the process of one validation class calling another validation class to perform additional validation checks.

- Call `validate()` method which in turn calls other methods.
- Invoke validate method on another validation class (may have to loop through a set of objects).
- Classes chain to validations of entities they hold.
- Refer to `PolicyPeriodValidation` class for examples of validation chaining. It calls other validation classes, such as:

  `PolicyContactRoleValidation`,

  `PolicyContactRoleForSameContactValidation`,

  `PolicyLocationValidation`, `AnswerValidation` and

  `PolicyLineValidation`.

Stop