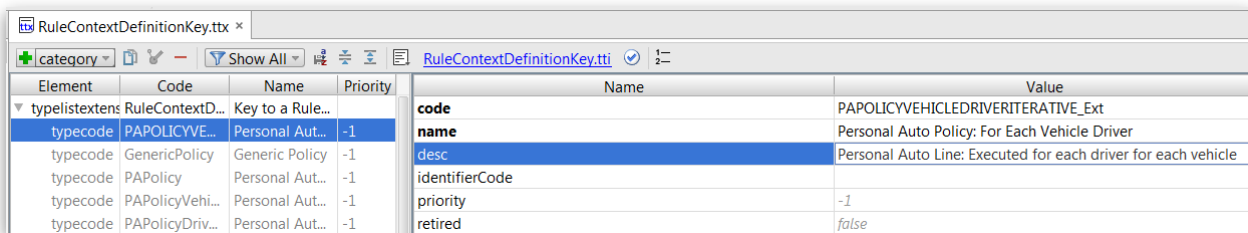


6.4 Solutions



Solution

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



Element	Code	Name	Priority
typelistextens	RuleContextD...	Key to a Rule...	
typecode	PAPOLICYVE...	Personal Aut...	-1
typecode	GenericPolicy	Generic Policy	-1
typecode	PAPolicy	Personal Aut...	-1
typecode	PAPolicyVehi...	Personal Aut...	-1
typecode	PAPolicyDriv...	Personal Aut...	-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	
priority	-1
retired	false

2. Add the new `RuleContextDefinitionKey` typecode to `UnderwriterEvaluatorTriggerPoint`.

```
11 class UnderwriterEvaluatorTriggeringPoint implements ITriggeringPoint {
12
27 override function supportedContexts(): Set<RuleContextDefinitionKey> {
28     return {
29         TC_GENERICPOLICY,
30         TC_PAPOLICY,
31         TC_PAPOLICYDRIVERITERATIVE,
32         TC_PAPOLICYVEHICLEITERATIVE,
33         TC_PAPOLICYVEHICLEDRIVERITERATIVE_Ext,
34
59     }
60 }
}
```

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.

```

35  uses gw.bizrules.provisioning.contexts.PersonalAutoVehicleDriversIterableUWContextDefinition

38  class BizRulesPlugin implements IBizRulesPlugin {
39  private final var _uwContexts: IRuleContextDefinition[]
40
41  construct() {
42  _uwContexts = {
43  new GenericUWRuleContextDefinition(),
44  new PersonalAutoUWContextDefinition(),
45  new PersonalAutoDriversIterableUWContextDefinition(),
46  new PersonalAutoVehiclesIterableUWContextDefinition(),
47  new PersonalAutoVehicleDriversIterableUWContextDefinition(),
48  .....
49  }
50  }
51  }

```

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

```

9  enhancement PolicyDriverEnhancement : entity.PolicyDriver {

144  property get YearsLicensed_Ext(): int {
145  var yearLicensed = (this.AccountContactRole as Driver).YearLicensed
146  var yearsLicensed = yearLicensed == null ? 0:
147  (Date.CurrentDate.YearOfDate - yearLicensed + 1)
148  return yearsLicensed
149  }

}

```

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.

Create New Rule

[Return to Underwriting Rules](#)

Rule Details

Advanced

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

BOP
Commercial Auto
CP
General Liability

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 VehicleDriver: \${paVehicleDriver.FixedId}

Value \${paVehicleDriver.AdditionalVersions.Year}

Short Description

English (US) Driver is newly licensed on vehicle \${paVe

Long Description

English (US) \${paVehicleDriver.PolicyDriver.DisplayNa

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).

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.

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

Risk Analysis

[Back](#)
[Next](#)
[Release Lock](#)
[Edit Policy Transaction](#)
[Versions](#)
[Bind Options](#)
[Close Options](#)

[Add UW Issue](#)
[Add Contingency](#)
[Request Approval](#)

[UW Issues](#)
[Contingencies](#)
[Prior Policies](#)
[Claims](#)
[Prior Losses](#)
[Motor Vehicle Records](#)

[Approve](#)
[Reject](#)
[Reopen](#)
[History](#)
 View Issues Blocking Me

Blocking Bind

<input type="checkbox"/>	Driver is newly licensed on vehicle 2004 Chevrolet Malibu in California	1	Approve	Reject
<input type="checkbox"/>	Driver is newly licensed on vehicle 2004 Chevrolet Malibu in California	3	Approve	Reject
<input type="checkbox"/>	Driver is newly licensed on vehicle 2003 Chevrolet Suburban in California	1	Approve	Reject
<input type="checkbox"/>	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:

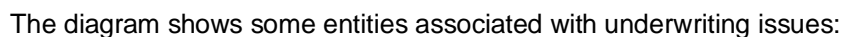
Underwriting rules conditions written using the Underwriting Rule screen can be viewed by authorized users in PolicyCenter. These rule conditions can be edited, enabled, and disabled, and can be efficiently moved between PolicyCenter instances. Underwriting rule conditions written in Gosu are not accessible 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.

6.5.3 UWIssue object model



- **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.

