

fintech **OS**

Insurance Product Factory 4.5.0

User Guide

TOC

Overview	7
Installing Insurance Product Factory	10
Prerequisites	10
Installation Steps	10
Upgrade	12
Post-Installation Setup	12
Creating Insurance Products	14
Create A New Insurance Product	15
Before Adding Your First Product	16
Main Info	17
Insured Object Type	28
Insured Object Type Read-Only Form	29
Product Coverages	30
Main Coverages	32
Optional Coverages	39
Premium Amount	41
Add Insurance Product Formula	41
Add Insurance Coverage Formula	47
Underwriting	53
Add Insurance Product Underwriting Formula	53
Add Insurance Product Coverage Underwriting Formula	59
Test Calculations	65
Test Calculations View	66

Create A Test Scenario	67
Documents Management	72
History	74
Managing the Product Factory	75
Related to this topic:	75
Insurance Business Formulas	78
Bring Your Formulas To The Factory	79
Demo Formulas	80
Insurance Lines Of Businesses	81
Insurance Products	85
Insurance Products View	86
Product Operations	87
Rules Impacting Products	94
Product Life Cycle	94
States Descriptions	96
Product State Transitions	96
Insurance Perils And Conditions	98
Insurance Perils View	98
Insurance Peril Insert Form	100
Insurance Perils Operations	101
Insurance Product Types	102
Insurance Product Types View	102
Insurance Product Type Insert Form	104
Insurance Product Types Operations	106
Insured Object Types	108
Insured Object Types View	110
Insured Object Type Insert Form	111
Insured Object Types Operations	115

Product Interrogation History	117
Product Interrogation History View	118
Solution Configurations	120
Configuration FAQs	120
Insurance Product Factory Endpoints	121
Get Formulas Structures API	125
Get Premium Amount API	132
Get UW Rules Result API	138
API Calls History	143
FTOS_IP_ProductInterrogationHistory	144
System Parameter	144
Insurance Product General Operations	145
Server Side Library	145
FTOS_IP_InsuranceProduct_Operations	145
Endpoints	154
Server Side On-demand Scripts	155
Cloning Insurance Products	156
Endpoints	156
Server Side Script Libraries	157
Importing Insurance Products	163
Data Import Templates	164
Product Templates	165
Product Formula Templates	166
Insured Object Type Templates	166
Data Config Definition	166
Troubleshooting Aspects	169
Configuring Payment Types	170
FTOS_IP_PaymentType	170

Configuring Insured Object Types	171
Entity FTOS_IP_InsuredObjectType	171
Entity FTOS_IP_InsuredObjectTypeDimension	172
Entity Attribute	173
Endpoints	173
Server Side On-demand Scripts	174
Server Side Libraries	176
Configuring Lines Of Business	179
Form Driven Flows	179
On Demand Script	181
Endpoint	182
Insurance Formulas	182
Attaching Formulas	184
Attach Premium Calculation Formulas	184
User Journeys	185
On Demand Scripts	186
Endpoints	186
Attach Insurance Underwriting Formulas	187
User Journeys	187
On Demand Scripts	188
Endpoints	188
Server Side Library	189
Defining Premium Splits	196
User Journey	197
Business Workflow Configuration Actions	197
Server Side Library	198
Mapping Formulas	199
User Journey	199
On Demand Scripts	199

Scripts For Product Formulas	200
Scripts For Product Items Formulas	200
Endpoint	201
Server Side Library	201
Testing Formulas	203
User Journey	203
Default Form Functions	204
Endpoints	212
On Demand Scripts	213
Server Side Script Library	216
Demo Formulas	218
Formula 1 - Bankassurance_UW_Formula_Final	218
Formula 2 - EmbededBankAssurancePremiumAmountFormula	219
Formula 3 - HomeServiceBankAssurancePremiumAmountFormula	220
Formula 4 - PA_Final_Premium	220
Formula 5 - PropertyFormula	221
Formula 6 - PropertyFormula_UW	223
Digital Assets	224
Glossary	228

Overview

Insurance Product Factory is an end-to-end solution for growing and managing your digital portfolio where you can create new insurance products, modify, or retire insurance products. As it evolves, the solution enables you to keep your digital portfolio disciplined, accessible, and comprehensible, while solidly handling the different life cycles of all your insurance products.

In conjunction with other **FintechOS** capabilities, you can **replicate your product data** into the **Insurance Product Factory** solution. Once you finished the bulk importing of product data, you are free to use your product knowledge to decide the degree of similarity between the old and the new products, to maybe create hybrid products, and also you have tools in place to test your creations.

Insurance Product Factory helps you with shortening the time from formula to product design, since it provides access to your formulas and a testing functionality without leaving the context of your product.

The **Insurance Product Factory** is about creating insurance products, as well as managing the product portfolio. See details about these functionalities in the [Creating New Insurance Product](#) and [Managing The Product Factory](#) pages.

Integrations

Insurance Product Factory can be integrated with other [insurance solutions](#) or [FintechOS automation blocks](#), allowing you to reap the resulting digital synergy. Few examples include:

- Leveraging the capabilities of [Business Formulas](#) to implement complex decision modeling for insurance peril rules and premium calculations, and apply them to different collections of insurance products, or product coverages. For more details, see also the [Insurance Business Formulas](#) page, in this guide.
- Using the [Proposal Configurator](#) solution on top of the **Insurance Product Factory**, in order to deliver a fully digital **Quote Configurator** customer experience - namely, allowing the eligible customers to review different insurance products, offers, or modules and configure

their own insurance quote.

- Using the [Digital Journeys](#) functionality to expose your products to your potential customers.
- Using different insurance [accelerators](#) on top of the **Insurance Product Factory** solution, in order to speed up the product delivery for specific insurance verticals.

More than that, for complex delivery projects, it is worth mentioning that the **Insurance Product Factory** can be also used in conjunction with different automation processors, that help insurers build their operations around the needs of their customers and have in-depth control over the product reach - such as [Omnichannel Campaigns](#), or [Hyper-Personalization Automation](#), and others.

Business Pain Points

FintechOS clients use the **Insurance Product Factory** module to respond to different challenges related to:

- Time-consuming routines;
- Routines more prone to human error when done manually;
- The sheer volume of changes to be recorded and accounted for;
- Little time to create insurance products and launch them;
- Pressure to adapt constantly to fluctuating markets.

Insurance companies use **Insurance Product Factory** to reduce the completion time required to create, configure, activate, change, and manage insurance product portfolios.

Insurance Product Factory Key Features

The **Insurance Product Factory** module enables insurers to achieve efficiency, gain flexibility and organize their insurance products data for analysis and operational purposes easily. The solution has the following key

features:

- Specific product configurations regarding availability periods for products, policy adjustments, policy transitions automated workflows or statement generation settings;
- Adjustable collections of insurance product coverages, optional coverages, modules and perils covered by the insurance products;
- Reduces completion time for premium amount calculations and underwriting scoring by integration with [Formula Engine](#).
- Product-related documents management;
- Full versioning functionality;
- Varying levels of access concerning users' involvement with a specific product, or product view;
- Safe: **FintechOS** role-based security architecture allows users, once authenticated, to interact only with the selected product, flow steps or interfaces associated with their role.

Installing Insurance Product Factory

Follow the instructions below to install and configure **Insurance Product Factory v4.5.0**. This is the first package to be installed in the entire insurance suite.

NOTE If you already have the previous version of Insurance Product Factory installed on your environment, follow the instructions from the [Upgrade](#) section.

Prerequisites

Before installing or upgrading to **Insurance Product Factory v4.5.0**, make sure the following are already installed:

- **HPFI v22.1.1.0**
- **SySDigitalSolutionPackages v21.1.0003**

Installation Steps

1. Unzip **Insurance Product Factory 4.5.0.zip** archive file.
2. Create a folder and name it **01 DeploymentPackages**.
3. Copy the **DigitalAsset** folder (if it exists) and **DigitalSolution M.N.U.P.RC.zip** file (extracted at step 1) inside the **01 DeploymentPackages** folder.

4. Locate the **FtosSysPkgDeployer** folder in the FintechOS installation kit (the path is <unzipped_install_archive>\Tools\FtosSysPkgDeployer). You need it to install the SySDigitalSolutionPackages.
5. Select and copy the **FtosSysPkgDeployer** folder.
6. Create the **install_Syspack.bat** file needed for installation.
7. Add the following script in the file and save it next to the *FtosSysPkgDeployer* folder.

```

CD /D %~dp0
"%~dp0\FtosSysPkgDeployer\FtosSysPkgDeployer.exe" -i -a -s
"StudioLink" -u AdminStudioUser -p user_password -z
DataBaseServer -v DB_user -k DB_user_password -d
"TheNameOfTheDataBase" -r "%~dp0\01 DeploymentPackages\*.zip"
Pause

```

8. Run the **install_Syspack.bat** script by double clicking on it.

Optionally, if you want to work with our demo insurance products, install **Insurance Product Factory Import Formula v4.4.0** and **Insurance Product Factory Import v4.4.0**, following the standard installation steps using, using the FtosSysPkgDeployer.

install_Syspack.bat file script parameters

- <StudioLink> - The web URL of the Innovation Studio installation, for example http://localhost/ftos_studio.
- <AdminStudioUser> - The username of the Innovation Studio user under which this import is executed. The user has to exist in Innovation Studio prior to this operation.
- <user_password> - The password for the Innovation Studio user.
- <DataBaseServer> - The name of the database server where the FintechOS installation database was created.
- <DB_user> - The username of the SQL Server user with administration rights on the FintechOS installation database.

- <DB_user_password>- The password for the above mentioned SQL user.
- <TheNameOfTheDataBase> - The name of the database where the Insurance Product Factory v2.4.0 is deployed.
- <syspack_path>- The physical path to the unzipped Insurance Product Factory v4.5.0 previously downloaded.

Upgrade

If you already have the Insurance Product Factory Import v4.4.0 package installed, upgrade to v4.5.0 by performing the below steps:

1. Check if near the **01 Deployment Packages** folder the **Upgrade** folder exists.
2. Run the SQL scripts located in the **Upgrade** folder.
3. Perform the installation steps above, steps 1-9, to install **Insurance Product Factory v4.5.0**.
4. Perform the same to install **Insurance Product Factory Import Formulas V4.5.0**. Run the SQL scripts from **Insurance Product Factory Import Formulas** folder.

Post-Installation Setup

After installing perform the following configurations:

1. Modify the app-settings Studio Vault key by adding the .xls value to the FileUploadWhiteList key:

```
{  
  "FileUploadWhiteList":  
  ".pdf,.doc,.docx,.els,.jpg,.jpeg,.xlsx,.dll,.ppt,.pptx,.txt,  
  .png,.ttf,.xml,.xls",
```

}

Creating Insurance Products

Feature Overview

The **Insurance Product Factory** helps you to evolve a diversified **digital collection** of re-usable insurance components - such as types, perils, or product coverages, that you further utilize as **building blocks** for the assemblage of highly personalized insurance products. A **product coverage** can be configured to have its own charging and underwriting structure, and a product can include multiple coverages, with different configurations. No coding skills required!

With the solution, you can create an unlimited number of insurance products by configuring insurance types, product coverages, perils, premiums, fees, conditions; and also by specifying the coverage, conditions for generating invoices, limit the availability of your products and more.

This solution uses **FintechOS Business Formulas** to help you reduce completion time for premium amount calculations, insurance peril scoring and underwriting evaluations, or for testing different product prices. For example, you can configure the system to calculate the premium at coverage level - this is especially useful in products that combine multiple Lines Of Business (LOB), or buckets of portfolios within the same LOB but with different peril profiles. For more details, consult also the **Insurance Business Formulas** section, from the [Managing The Product Factory](#) page.

Once built, your products can be edited, versioned, cloned, multiplied. When activated, a product becomes digital journey-ready, that is you can embed it into different digital journeys and expose it to your potential customers, through different digital touchpoints, channels, or portals.

Create A New Insurance Product

To create an insurance product by using the **Insurance Product Factory** wizard, in **Innovation Studio**, follow the steps below:

1. In **Innovation Studio**, at the top left corner of your screen, click the main menu icon to open the main dropdown list.
2. From this main list, click **Insurance Product Factory**. A second panel opens, to the left.
3. Next, click **Insurance Products** to go to the **Insurance Products List**.
4. On the **Insurance Products List** page, click **Insert**, at the top right corner of the page, to add a new record. The **Insurance Product** dynamic form opens.
5. Use the **available product configurations tabs** listed below, for creating your product:
 - [Main Info](#) - You start from here! Read also the [Before Adding Your First Product](#) instructions.
 - [Insured Object Type](#)
 - [Product Coverages](#)
 - [Premium Amount](#)
 - [Underwriting](#)
 - [Test Calculations](#)
 - [Documents Management](#)
 - [History](#)

Below, you can see the product creation wizard, for an example property **Product**, in **Draft** status:

innovation studio Administrator EN

CURRENT STATUS: Draft NEXT STATUS: Choose status

CURRENCY: EUR TYPE: Home NAME: Household Star Insurance TSLC1 START DATE: 14/02/2022

Save and reload Business Transactions

1 Main Info 2 Insured Object Type 3 Product Coverages 4 Premium Amount 5 Underwriting 6 Test Calculations 7 Documents Management 8 History

General Data

Insurance Product Type: Home Insurance Product Code: TSLC1

Name: Household Star Insurance TSLC1 Currency: EUR

Start Date: 14/02/2022 End Date: 01/07/2025

Maximum Discount (%): 30 Maximum Commission (%): 20

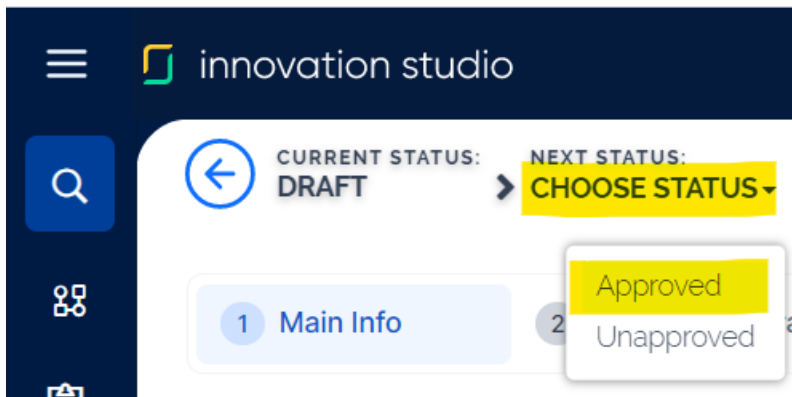
Description:

Commercial Text:

Before Adding Your First Product

- The dynamic form allows you to create your desired product and configure the product settings.
- To complete the creation journey for any product, **you go through all the tabs listed above.**
- Filling in information in the first tab **activates the next tabs** and lets you move forward.
- Having at least one **Insurance Product Type** defined and one **Insured Object Type** approved, helps you move faster through the creation journey. However, you also have the option to insert these details when configuring the product, instead of picking them from a list of existing items.
- The wizard displays different product configuration options, **based on what you establish** as product behavior when filling in information **in the first tab.**
- If you need to interrupt your journey, you can use the **Save and close** button to save your product as **Draft**. While the product is in **Draft** business status, the form is still editable, and you can complete the journey at a later time.

- Keep the product in **Draft** status until you are confident the design is finished. A **Draft** product is always editable. So, you can come back to edit it, as many times as needed.
- Once a product is in **Approved** business status, it cannot be edited the same way the **Drafts** products can be. For editing an **Approved** product, you must create a new version of it. For that, you use the versioning (+) button, at the top right corner of the page. For more details, go to the **Versioning Products** section, on [Product Operations](#) page. After adding a new version to your product, you must **manually approve** it.
- After you finished configuring it, you must **activate your product** by using the status picker, at the top left corner of your product screen. Below, an example of the status picker:



HINT

Product **status** determines if your insurance product is used in digital journeys targeting potential customers. In order to make it available to different digital journeys, the product should be in the **Approved** status. For more details, see the [Product Life Cycle](#) page.

Main Info

The **Main Info** tab lets you add information about your product, and also indicate some of its underlying business conditions. The tab has two sections:

- **General Data** - This section lets you introduce the product's main characteristics, those which are most visible to the final customer.
- **Business Process Configuration** - This section lets you configure the policy coverage, policy administration, the scheduling of payments and billing, the management of claims, the tariff type, and more.

NOTE

Filling in information in this first tab **activates the next tabs** and lets you move forward.

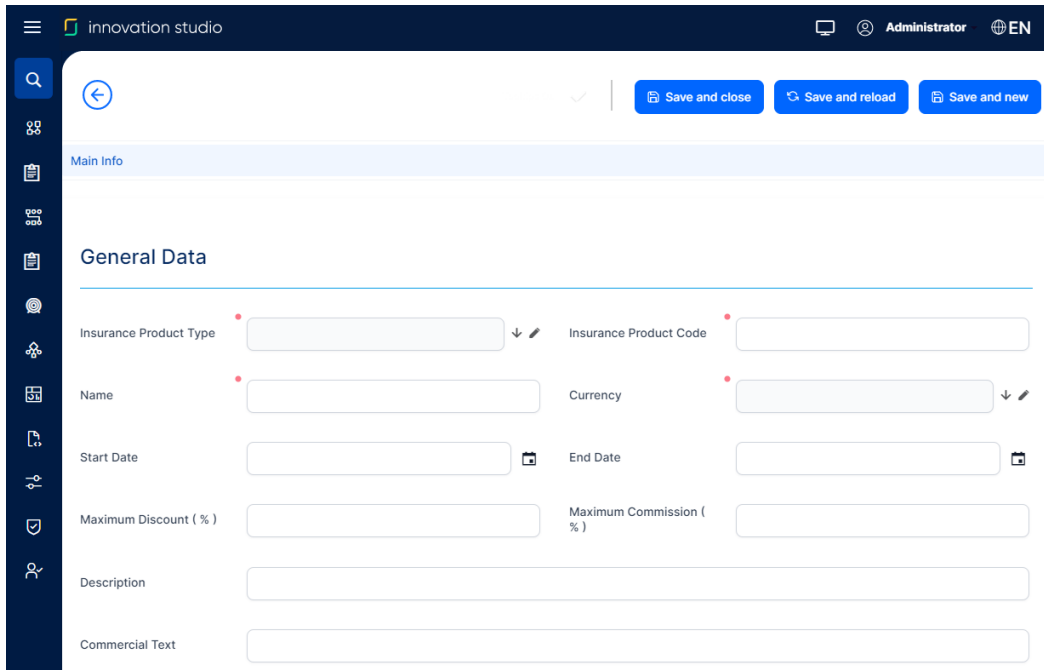
Additionally, **what you configure in this tab has impact on what fields you see on some of the other tabs**. However, after filling in information in the **Main Info** tab, it is not absolutely necessary to work on the other tabs in the order listed on your screen. For example, after the **Main Info** tab, you might want to go to the **Documents** tab, to upload the documents of your product, if you already have them. You can do that.

IMPORTANT!

Understanding **default** vs **specific** settings, when configuring products:
A default setting uses the default values from the Insurance Parameter set up in your **Portal > Settings > Insurance parameters**. A specific setting uses values which differ from the default ones - values that you only configure for that specific field, section, or product.

General Data Section

Below, an example of the configuration fields available in the General Data Section - the red dots mark the mandatory fields:



In this section, generally describe your new product. The following fields are available:

Field	Description
Insurance Product Type	Click the dropdown to select a Type of insurance for your insurance product - ex. Auto, Health, Home, Travel. See Insurance Types for details.
Insurance Product Code	Use this text area to fill in the code of the insurance product.
Name	Insert the name of your insurance product.
Currency	From the dropdown list, select a currency for your insurance product.
Start Date	Pick the date when your product becomes available.
End Date	Pick the date when the availability of your product ends.
Maximum Discount (%)	Set the maximum percentage of the commercial discount which can be offered in the sales process for your product. This field allows numeric values.
Maximum Commission (%)	Set the maximum percentage of the commission which can be offered to intermediary sellers. This field allows numeric values.
Description	Field for describing the insurance product (500 characters).
Commercial Text	Field for commercial text (300 characters).

After inserting the required information, move to the next section of the tab.

Business Process Configuration

Below, an example of the available configuration fields, with some details inserted, for a **Draft** product:

The screenshot displays the 'Business Process Configuration' interface. It is organized into several sections:

- Policy Coverage:** Includes a 'Total Indemnity Limit' field.
- Policy Admin:** Includes a 'Free Withdrawal Period Limit Days' field (value: 25).
- Renewal Type:** A dropdown menu currently set to '[None]'.
- Prorate Type Configuration:** A dropdown menu currently set to 'Default'.
- Payments Schedule & Billing:** Includes a 'Payment Period Grace (days)' field (value: 15) with a 'Write Off' checkbox, and a 'Specific Write-off' dropdown menu.
- Premium Invoice Generation:** A dropdown menu currently set to 'Default'.
- Claims Management:** Includes a 'Claim Notification Period Limit (hours)' field (value: 48).
- Update Indemnity Limit:** A checkbox that is checked.
- Tariff Configuration:** Includes a 'Tariff Type' dropdown menu (value: 'Per Coverage') and an 'Underwriting Type' dropdown menu (value: 'Per Coverage').

In this section, the following fields are available:

Policy Coverage

Field	Description
Total Indemnity Limit	Insert the maximum coverage amount provided per insurance policy.

Below, an example of the available policy configuration fields (see also the next section):

Business Process Configuration

Policy Coverage

Total Indemnity Limit

Policy Admin

Free Withdrawal Period Limit Days

Renewal Type

Prorata Type Configuration

Policy Administration

Field	Description
Free Withdrawal Period Limit Days	Set the limit (expressed in days) for the free withdrawal period, if any. This value is necessary for the cancellation processes.

Field	Description
Renewal type	<p>Choose the type of the insurance policy renewal. The option set values are: [none], No (default), Automatic renewal, and Renewal offers.</p> <p>If you choose Automatic renewal, the following option set fields become available:</p> <ul style="list-style-type: none"> - Renewal Validity - where the options are: Yearly, Monthly, and SameValidity. - Renewing Policy - where the options are: [none], Same Policy and New Policy. - Renewal Tariff - where the options are: Same tariff and Actual tariff. - No of Days Before Renewal - Insert the number of days before policies reach maturity that the system can use to notify you about the coming renewal opportunity.
Prorata Type Configuration	<p>Set the proportion rate type for premium payments. The option set values are:</p> <ul style="list-style-type: none"> - Default if you allow the generic setting to be used. - Specific if you want to set a specific rate. When you choose this option, the Prorata Type option set becomes available and you can choose between the following specific values: Daily and Monthly.



Payments Schedule & Billing

Below, an example of the available configuration fields:



Payments Schedule & Billing

Payment Period Grace (days)

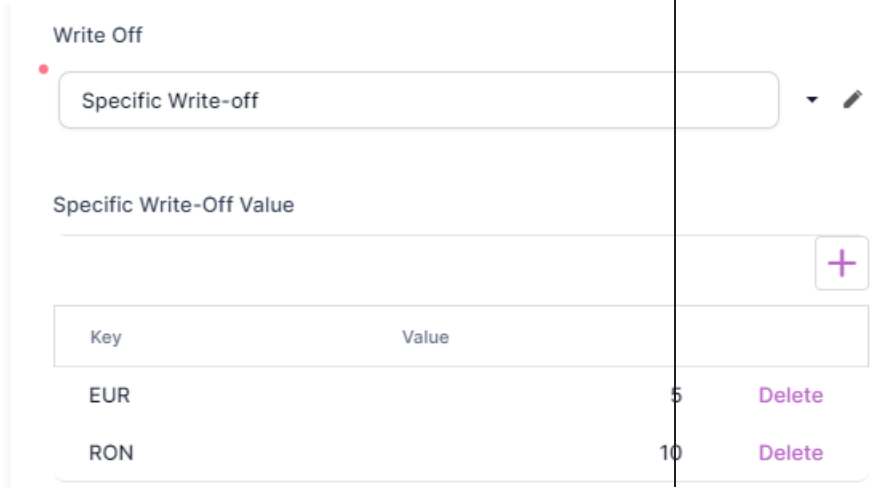
Write Off

• Default  

Premium Invoice Generation

• Default  

Field	Description
Payment Period Grace (days)	Insert the number of days for the payment grace period.

Field	Description
Write off	<p>Set the tolerance threshold for writing off payments that fall shorter than the expected agreed installment amount (for example 74.5\$ instead of 75\$). The option set values are: No write-off, Default, and Specific Write-off.</p> <p>Choose:</p> <ul style="list-style-type: none"> - No write-off if no rule for writing off is applied for your product. - Default if you allow the default write-off settings to be used. - Specific Write-off if you want to set the value for this parameter yourself. When you choose the Specific Write-Off option, a Json pop-up appears and you can add your own key-value pairs, per different currencies. Set the desired values for this parameter. The values you insert are automatically saved. <p>Below, an example of the Json pop-up, with some added values:</p> 

Field	Description
Premium Invoice Generation	<p>Configure the values to be used for automatic premium invoice generation. The option set values are: Default, Specific SGDAY* and Specific Day. Choose:</p> <ul style="list-style-type: none"> - Default if you allow the default setting to be used for automatic invoice generation. - Specific SGDAY if you want to set a specific day for invoice generation. When you choose this option, the No. of Days in Advance (SGDAY) field becomes available and you can insert a specific number. - Specific Day if you want the automatic invoice generation to be performed by the system on a specific day of the month. When you choose this option, the Specific day of the month field becomes available and you can specify a day of the month for generating the invoice.

***SGDAY** translates to Statement Generation Day - This parameter stores the number of days in advance (before the payment due date), for automatically generating an invoice (statement) for a scheduled payment (installment) on a policy.

Click **Save and reload**.

The form shows the **Main Payment Type** insert form, as in the example below:

Main Payment Type

Payment Type	Main Payment Type
<input type="text" value="Q"/>	<input type="text" value="(All)"/>
PayU-on time	<input type="checkbox"/>
Bank transfer	<input type="checkbox"/>
Direct Debit	<input type="checkbox"/>
Broker Collection	<input type="checkbox"/>
PayU	<input type="checkbox"/>

The available payment types are loaded into the grid and you must click next to the type that you choose as **Main Payment Type**. A message appears to let you know that the chosen Main Payment Type is saved.

NOTE

The payment types already available as option set values are: **OP** (bank transfer), **PayU**, **PayU-on time**, **brokerCollection**, and **directDebit**. The default value is **none**. If necessary, the payment type grid allows you to add new payment type values into the grid, or delete those that don't fit your needs.

Claims Management

Below, an example of the available configuration fields:

Field	Description
Claim Notification Period Limit (hours)	The period during which the claimant can notify the damage produced - in order to open a claim on a policy that contains this insurance product, expressed in hours.
Update Indemnity Limit	Check the box in order for the system to make automatic updates of the indemnity limit, after every claim payment.

Tariff Configuration

You can use two types of rating models:

1. **Top down** - where pricing and underwriting is made at **product level**.

Per product pricing means you add a generic formula on product level, no matter how many coverages, or riders it has. For example, for a health insurance product, the customer receives a policy (with a list of coverages and riders, that appear on the policy), but no matter of what is included in the policy, the customer will pay 50 Eur/ month if their age < 60 years and 60 Eur/ month if their age is >= 60. So, the premium is not calculated for each coverage, it's a general price, calculated on product level.

2. **Bottom up** - where pricing and underwriting can be set from peril (risk), or **coverage level** and, then, they can be aggregated at product level. Use this approach to attach individual formulas per each product item (coverage).

The available fields are as follows:

Field	Description
Tariff Type	The tariff type options. The option set values are: Per Coverage and Per Product .
Underwriting Type	The underwriting type options. The option set values are: Per Coverage and Per Product .

Below, an example of the available configuration fields:

The screenshot shows a configuration window titled "Tariff Configuration". It contains two sections, each with a red error indicator (a small red dot) to its left. The first section is labeled "Tariff Type" and features a dropdown menu currently displaying "Per Coverage". To the right of the dropdown is a small pencil icon for editing. The second section is labeled "Underwriting Type" and also features a dropdown menu displaying "Per Coverage" with a pencil icon to its right.

After configuring all of the above, click **Save and reload**.

The other tabs become available and you can see them at the top of the screen.

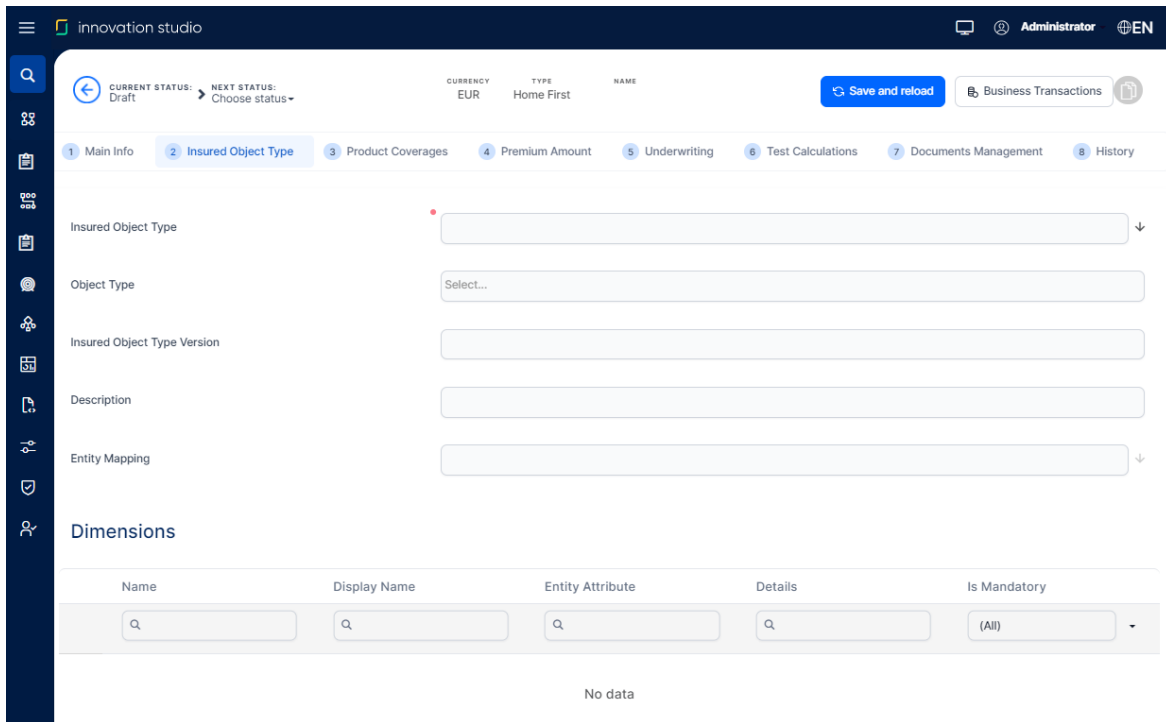
HINT

Continue to the [Insured Object Type](#) tab.

Insured Object Type

The **Insured Object Type** tab lets you attach an object to a product - either by choosing one from your **Insured Object Types List** or by inserting a totally new one. After adding the object, you can see all the object dimensions, in read-only format. For more details about creating, editing, or versioning objects, consult the [Insured Object Types](#) page.

Below, you can see an example of a blank **Insured Object Type** insert form:



Follow the below steps to embed an object into your product:

1. In the **Insured Object Type** tab, locate the **Insured Object Type** field, in the header of the form.
2. Click the arrow icon, to the right, to expand the dropdown list. This list displays all your **Approved** objects.
3. From the list, select an object and click **Ok**.
4. Next, click **Save and reload**. All the object details, including its **Dimensions**, are loaded into the form and you can see them in read-only format.
5. If you need to change the object (while your product is in **Draft** status), repeat the exact same steps.

IMPORTANT!

In order to approve a product, it must have an **object type** attached!

Insured Object Type Read-Only Form

After **Insert**, and **Save and reload**, the **Insured Object Type** form displays the following read-only fields:

Insured Object Type Section

Field	Description
Insured Object Type	The name of the object.
Object Type	The type of the object - either motor, property, travel, or personal accidents .
Insured Object Type Version	The object's logged version number.
Description	The description of the object.
Entity Mapping	The entity mapping.

Dimensions Section

Inside this grid you can see all the dimensions associated with your object.

Below, an example of the **Insured Object Type** read-only form, for a property **Object**:

Name	Display Name	Entity Attribute	Details	Is Mandatory
addressId	addressId	addressId	addressId	<input type="checkbox"/>
alarmSystemConnectedToSecu...	alarmSystemConnectedToSecu...	alarmSystemConnectedToSecu...	alarmSystemConnectedToSecu...	<input type="checkbox"/>
annexSurface	annexSurface	annexSurface	annexSurface	<input type="checkbox"/>
ApartmentNo	ApartmentNo	ApartmentNo	ApartmentNo	<input type="checkbox"/>

HINT
Continue to the [Product Coverages](#) tab.

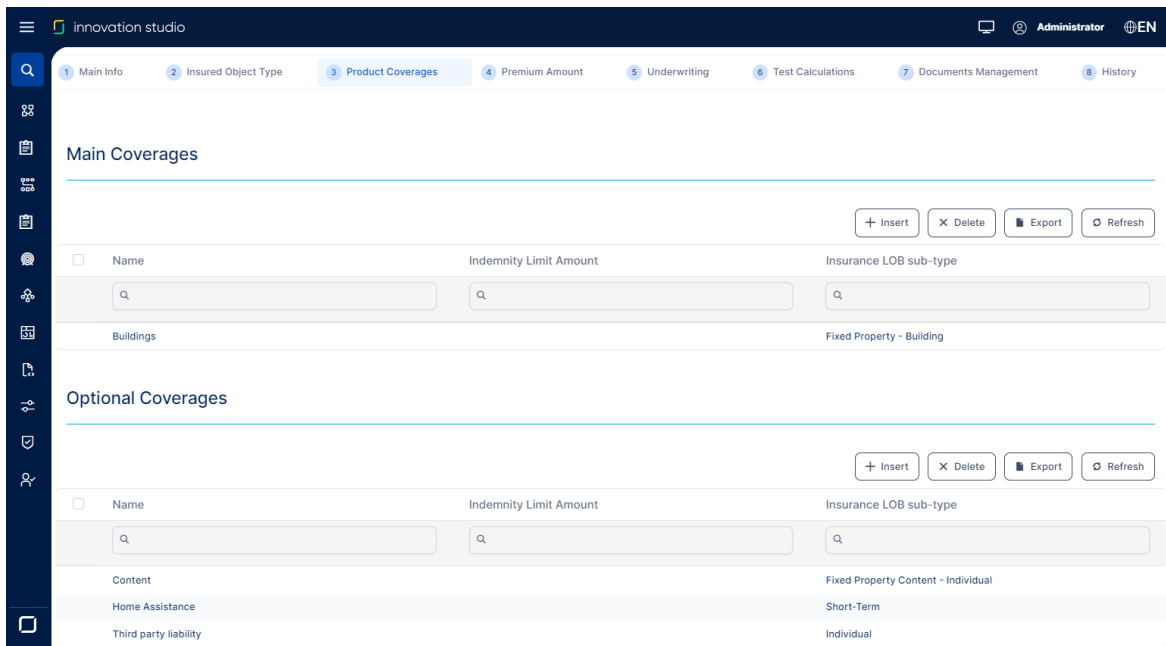
Product Coverages

The **Product Coverages** tab allows you to attach coverages to your insurance product. For example, for property insurance, a customer might buy insurance that has two types of coverages: one for the house and the other one for the contents of the house.

The **Product Coverages** tab has two sections:

- **Main Coverages** - This section is reserved for **Base** coverages. For example, for a **Property Insurance** policy, the **Base** insurance product item could be the coverage for the actual building, only. And the coverage for the contents of the house could be included as a **Rider** - an optional insurance product item, that would be charged separately.
- **Optional Coverages** - In this section, you can include **Riders** - additional perils that the customer wants to cover. For example, for a Life Insurance policy, additional coverage may potentially refer to losing working capacity.

Below, an example of the available configuration sections, with some attached coverages, for a **Draft** product:



Depending on your product, the **Main** and **Optional Coverages** grids can display several coverages. If needed, you can use the **Search by** option at the top of every column, in order to find a certain coverage record. The available columns (search types) are: **Name**, **Indemnity Limit Amount**, and **Insurance LOB sub-type**.

Main Coverages

The **Main Coverage** insert form allows you to configure any number of main coverages for your insurance product. It also lets you attach all necessary documents describing each added coverage.

Below, an example of the **Main Coverage** insert form:

The screenshot shows the 'Insurance Product Coverage' form within the 'innovation studio' application. The form is divided into two main sections: 'Main Coverage' (active) and 'Documents'. The 'Main Coverage' section contains the following fields:

- Insurance Product:** A dropdown menu with a search icon.
- Name:** A text input field.
- Waiting Period:** A text input field.
- Indemnity Limit:** A text input field.
- Indemnity Percentage:** A text input field.
- Commercial Description:** A large text area.
- Line of Business Subtype:** A dropdown menu with a search icon.
- Code:** A text input field.
- Waiting Period Type:** A dropdown menu with 'Select...' as the current value.
- Indemnity Limit Currency:** A dropdown menu with a search icon.

At the top right of the form, there are three buttons: 'Save and close', 'Save and reload', and 'Save and new'. A left sidebar contains various navigation icons, and the top header shows 'innovation studio', 'Administrator', and 'EN'.

1. Insert Main Coverage

In the **Main Coverages** grid, click **Insert**.

The screenshot shows the 'Main Coverages' grid. At the top right, there are four buttons: '+ Insert' (highlighted in yellow), 'X Delete', 'Export', and 'Refresh'. Below the buttons is a table with the following columns:

- Name:** Includes a search icon and the value 'Buildings'.
- Indemnity Limit Amount:** Includes a search icon and is currently empty.
- Insurance LOB sub-type:** Includes a search icon and the value 'Fixed Property - Building'.

The **Insurance Product Coverage** window opens.

The following fields are available for configuring a **Main Coverage**:

Field	Description
Insurance Product	The Insurance Product that includes the current main coverage. This information is automatically filled in by the system.
Line of Business Sub-type	From the dropdown, select the LOB Sub-type for the main coverage. For configuration details, see the Lines Of Businesses page.
Name	Add a name for the main coverage.
Code	Insert a code for the main coverage.
Waiting Period Type	From the option set, choose Days , Weeks or Months to indicate the type of waiting period.
Waiting Period	Insert the number of units for the waiting period (configured in the Waiting Period Type option set), until the current coverage becomes available on a policy.
Indemnity Limit	The maximum monetary amount provided on the policies incorporating this main coverage.
Indemnity Limit Currency	From the dropdown list, select a currency for the indemnity limit.
Indemnity Percentage	You can set the main coverage indemnity limit as a percentage from the total indemnity limit of the insurance product.
Line of Business Sub-type	From the dropdown, select the LOB Sub-type for the main coverage. For configuration details, see the Lines Of Businesses page.
Commercial description	Text area for describing the main coverage.

Click **Save and reload**. Next, the **Sub-coverages** grid becomes available.

Continue to the next section.

2. Insert Sub-Coverage

The **Sub-coverage** grid allows you to append perils (e.g. natural disasters), or groups of perils, to the main coverage.

Below, an example of the **Sub-coverage** grid, available at the bottom of the **Main Coverage** insert form:

Insurance Product Coverage

Insurance Product: Household Star Insurance TSLC1 | Line of Business Subtype: Fixed Property - Building

Name: Household Insurance | Code: FPB145

Waiting Period: | Waiting Period Type: [none]

Indemnity Limit: | Indemnity Limit Currency: | Indemnity Percentage: | Commercial Description:

Sub-coverages

+ Insert | X Delete | Export | Refresh

Name | Item Type

🔍 | 🔍

In the **Sub-coverages** grid, click **Insert** to open the **General** window. This insert form allows you to attach all necessary sub-coverages, to the current main coverage.

Below, an example of the Sub-coverage insert form, with some filled-in details:

Innovation studio | Administrator | EN

Save and close | Save and reload

General

Sub-coverage

Name: Example Sub-Coverage | Code: ESC_123

Item Type: Group | Parent Coverage: Second Building

Insurance Type: Home Property

Icon: Add file | Drop file here | Nat-Cat Coverage:

Commercial Description: Hail

The following fields are available for configuring a Sub-coverage:

Field	Description
Name	Add a name for the Sub-coverage .
Code	Add a code for the Sub-coverage .
Item Type	It is automatically filled in by the system with the type Group .
Parent Coverage	It is automatically filled in by the system with the name of the Main Coverage .
Line of Business Sub-type	It is automatically filled in by the system with the current LOB type.
Icon	Upload an icon for the peril/ condition, if necessary.
Nat-Cat coverage	Check the box if the peril/ condition belongs to the Natural Catastrophes group.
Commercial description	Text area for describing the sub-coverage.

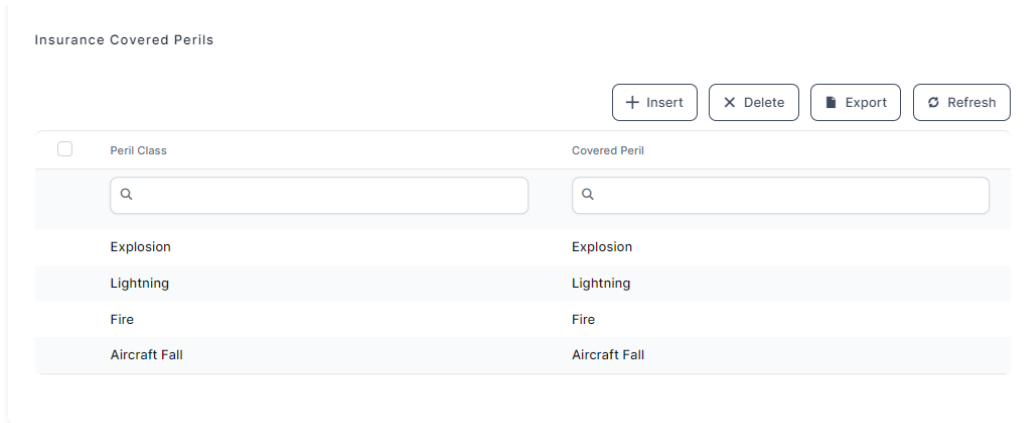
Click **Save and reload**. Next, the **Insurance Covered Perils** view becomes available.

Continue to the next section.

3. Validate Covered Perils

Adding details for a **Sub-coverage** (described above) unlocks a new view - the **Insurance Covered Perils** grid, at the bottom of the form, in which you can see a list of all the types of perils and conditions covered by that particular **Sub-coverage**. You can explore the list of perils and if necessary, you can edit this list, by pressing **Insert**, or **Delete**. You can also edit the **Peril Types** from the [Insurance Perils And Conditions](#) menu.

Below, an example of the **Insurance Covered Perils** grid, with some filled-in details:



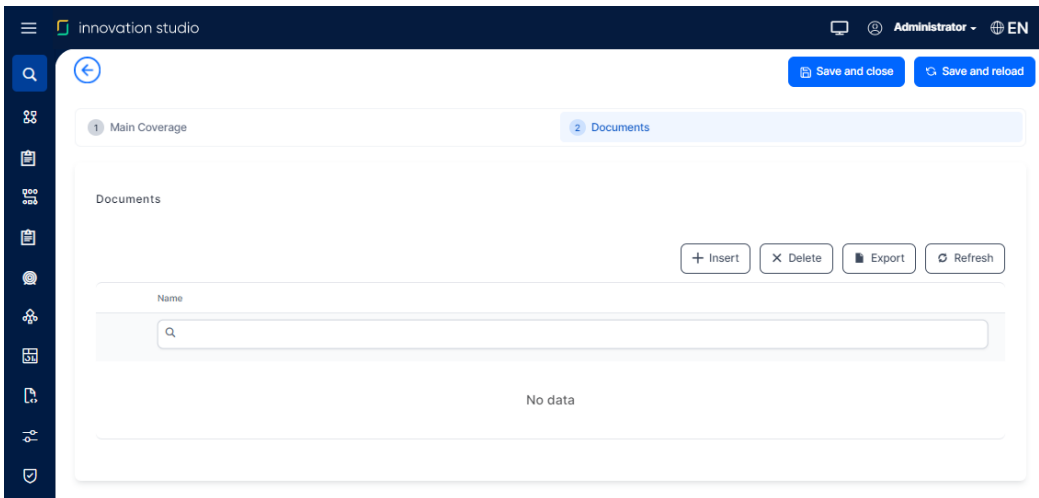
Click **Save and close**. You are directed to the **Main Coverage** insert form. From here, click the **Documents** tab.

Continue to the next section.

4. Add Insurance Documents

The **Documents** tab allows you to add any relevant documents related to your coverage.

Below, an example of the Documents Tab:



In the **Documents** grid, click **Insert** to open the **Add Document** window. This insert form allows you to attach all documents relevant to the current main coverage.

For each **Document** that you upload, the following information must be provided:

Field	Description
Name	The name of your document.
Display name	The display name of your document.
Document type	From the option set, choose between Policy, Terms & Conditions or IPID - Insurance Product Information Document - which is a type of document presented during the Quote&Apply flow.
Code	The code for your document.
Included in offer template	Check the box if your document must be included in the product offer template.
Included in the policy template	Check the box if your document must be included in the policy template.

Click **Add File** to upload your document. Repeat this step for each document that you need to attach.

5. Repeat

Add as many **Main Coverages** as you need for your insurance product. Use the **Sub-coverages** grid on every **Main Coverage** to attach perils and conditions for that particular coverage.

Below, an example of a Main Coverage, with some Sub-coverages, for a property insurance product:

Insurance Product Coverage

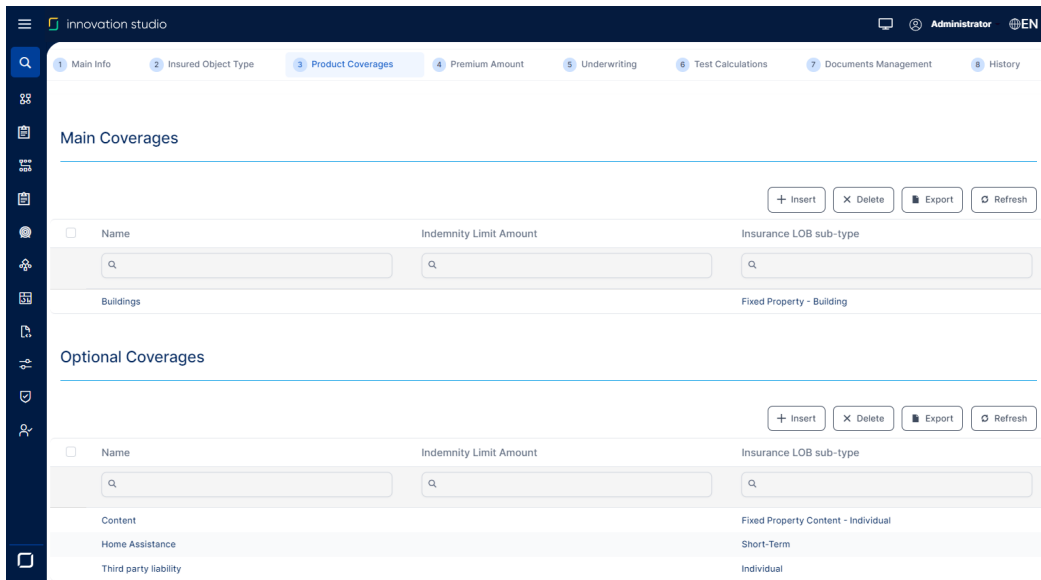
Insurance Product	<input type="text" value="Household Star Insurance TSLC1"/>	Line of Business Subtype	<input type="text" value="Fixed Property - Building"/>
Name	<input type="text" value="Household Insurance"/>	Code	<input type="text" value="FPB145"/>
Waiting Period	<input type="text"/>	Waiting Period Type	<input type="text" value="[none]"/>
Indemnity Limit	<input type="text"/>	Indemnity Limit Currency	<input type="text"/>
Indemnity Percentage	<input type="text"/>		
Commercial Description	<input type="text"/>		

Sub-coverages

Name	Item Type
Fixed Property - Building 1	Group
Fixed Property - Building 2	Group

Once finished, click **Save and close** and navigate back to the main **Product Coverages** window. Here, locate the **Optional Coverages** section and proceed to add optional (rider) coverages for your insurance product.

Below, an example of the **Product Coverages** tab:



If the case, continue to the **Optional Coverages** section, below.

Optional Coverages

The **Optional Coverages** insert form allows you to configure any number of optional coverages for your insurance product. It also lets you attach all necessary documents describing each added coverage. This form is similar to the Main Coverage insert form, described in the section above.

Below, an example of the **Optional Coverages** insert form:

1. Insert Optional Coverage

Under **Optional Coverages**, click **Insert**. The **Insurance Product Coverage** window opens.

The following fields are available for configuring each **Optional Coverage** (Insurance Product Item):

Field	Description
Insurance Product	The Insurance Product that includes the current optional coverage. This information is automatically filled in by the system.

Field	Description
Line of Business Sub-type	From the dropdown, select the LOB Sub-type for the main coverage. For configuration details, see the Lines Of Businesses page.
Name	Add a name for the optional coverage.
Code	Insert a code for the optional coverage.
Waiting Period Type	From the option set, choose Days , Weeks or Months to indicate the type of waiting period.
Waiting Period	Insert the number of units for the waiting period (configured in the Waiting Period Type option set), until the current coverage becomes available on a policy.
Indemnity Limit	The maximum monetary amount provided on the policies incorporating this optional coverage.
Indemnity Limit Currency	From the dropdown list, select a currency for the indemnity limit.
Indemnity Percentage	You can set the optional coverage indemnity limit as a percentage from the total indemnity limit of the insurance product.
Line of Business Sub-type	From the dropdown, select the LOB Sub-type for the optional coverage. For configuration details, see the Lines Of Businesses page.
Commercial description	Text area for describing the optional coverage.

Click **Save and Reload**.

2. Insert Sub-Coverage

Use the **Sub-coverages** grid to attach sub-coverages, and also perils, for the **Optional Coverage**. To do so, please follow the same steps as in described in the [Insert Sub-Coverage](#) section, above.

3. Validate Covered Perils

Use the **Insurance Covered Perils** grid to explore the list of perils and update it, if necessary. To do so, please follow the same steps as in described in the [Insurance Covered Perils](#) section, above.

4. Add Insurance Documents

Use the **Documents** tab to add any relevant documents related to the **Optional Coverage**. To do so, please follow the same steps as in described in the [Add Insurance Documents](#) section, above.

5. Repeat

If the case, repeat the same steps to add more **Optional Coverages** for your insurance product. Once finished, click **Save and close**.

HINT

Continue to the [Premium Amount](#) tab.

Premium Amount

The **Premium Amount** tab allows you to configure how premium amounts are calculated for your insurance product, or product items (coverages), by attaching calculation formulas to them. You can also change previous configurations by deleting attached formulas and inserting new ones, if needed.

Based on the **Tariff Type** - previously chosen in the **Tariff Configuration** section, on this tab you can either see the **Insurance Product Formula** grid or the **Insurance Product Item Formula** grid.

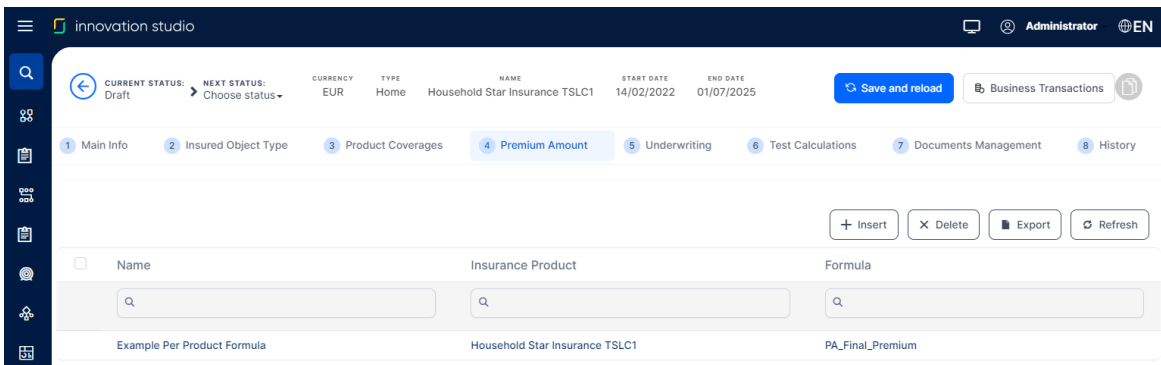
NOTE

Only one formula can be attached to a specified product, or product item (coverage), at a time.

Add Insurance Product Formula

Prerequisite: You must have previously selected the **Per Product** tariff type, in the **Main Info** product tab. This selection makes the **Add Insurance Product Formula** insert form available in the **Premium Amount** tab.

Below, an example of the insurance product with an attached insurance product formula:



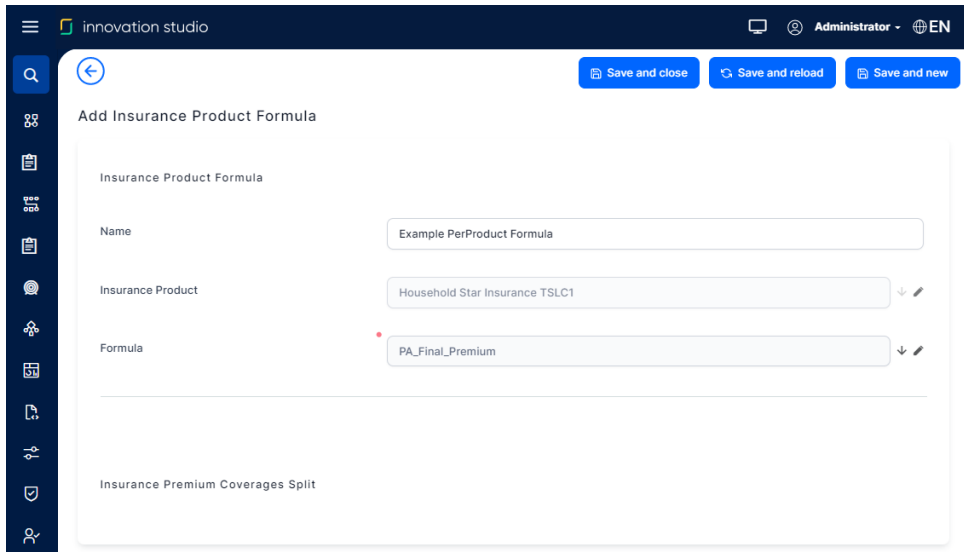
NOTE
 The insert form is editable. You can delete a configuration and attach some other formula - instead of the previously chosen one, if needed.

To add a formula that helps you calculate the premium amount per product, follow the steps below:

1. Create A Configuration

1. In the **Premium Amount** tab, click **Insert**.
2. The **Add Insurance Product Formula** insert form opens. Notice the name of the insurance product being auto-completed and not editable.
3. In the insert form, fill in a descriptive **Name** for the formula.
4. Use the picker (arrow) to select from the list a **Formula** to be appended to the current product for premium amount calculation.

Below, an example of the insurance product formula insert form, with some filled in details:

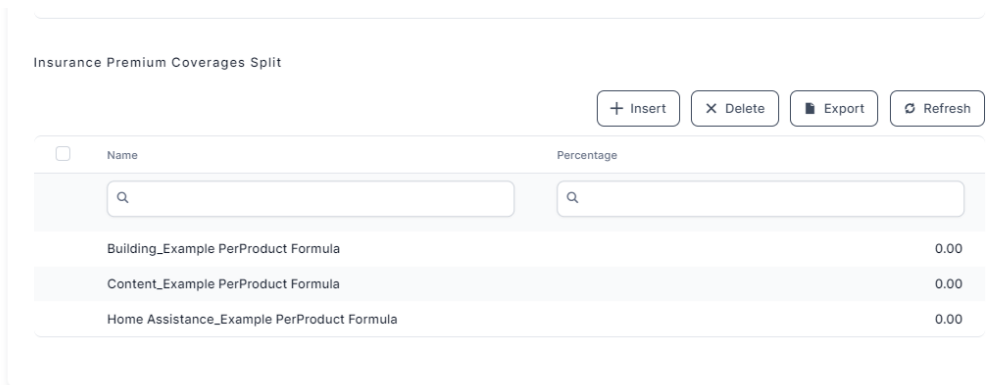


5. Click **Save and reload**.
6. Move to the next step, below.

2. Define The Coverage Split

After **Save and reload**, the **Insurance Premium Coverages Split** section becomes available, too - at the bottom of the form.

Below, an example of the coverages split grid, available after save and reload:



There are cases when the **Premium Coverage Split** is necessary for the premium calculation - for example when you make a **Proposal Configurator** API call in order to get prices for different coverages that might be interesting for customers looking for an insurance quote.

IMPORTANT!
The percentage for every coverage is 0 by default and, when not set, you cannot activate the product.

Proceed with indicating the premium coverage split, as follows:

1. Go to the **Insurance Premium Coverages Split** grid.
2. Next to each item from the grid, set the **percentage** used to calculate the premium amount per each coverage included in the product.

Below, an example of the **Insurance Premium Coverages Split** grid, with filled values:

Name	Percentage
Building_Example PerProduct Formula	60.00
Content_Example PerProduct Formula	20.00
Home Assistance_Example PerProduct Formula	20.00

3. Once finished, click **Save and reload**.
4. Move to the next step, below.

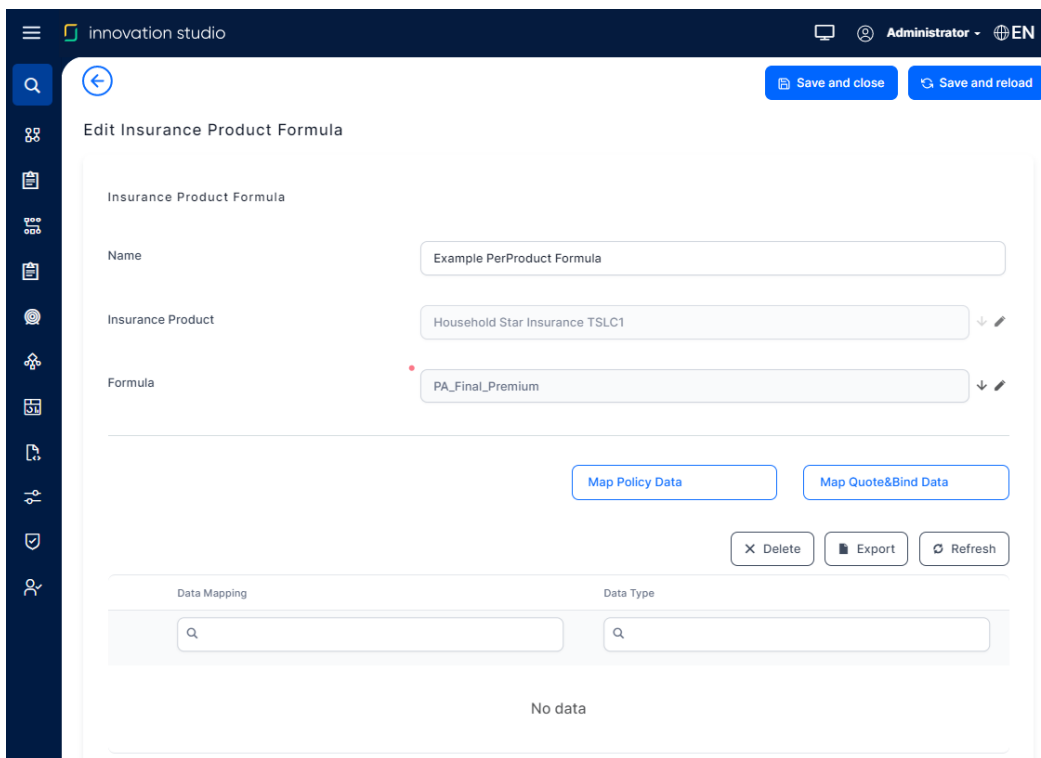
3. Map Data

IMPORTANT!
Mapping data is a mandatory step in order for the system to know from where to fetch the product data necessary for the **formula engine** to run.

After **Save and reload**, the **Map Data** buttons become available.

These buttons allow you to define a single data mapping, for either a **Policy Admin** or a **Quote and Bind** flow. When clicking either of the buttons, you can select a **Master Entity** (any master entity that you need, for your calculations). This way, you get access to use the values of the selected entity's attributes as **formula input keys**, to obtain a certain output (and this output is registered on some other entity).

Below, an example of the insert form for the insurance product formula - with **Map Data** buttons available, after save and reload:



Proceed with mapping the necessary data for the attached formulas, as follows:

1. Click the **Map Data** button - either **Map Policy Data** or **Map Quote&Bind Data**, depending on the mapping you need to perform. The **Formula Parameter Mapping** form opens.

2. In the **Definition** tab (first tab), notice the **Data Mapping Type**, the **Operation Name** (the actual formula name), the **Name** (the mapping name) fields being auto-completed and not editable.
3. Use the **arrow selector** to indicate a **Master Entity** from which the system will extract the input keys for the formula. .

Below, an example of a **Formula Parameter Mapping** form, with definition details:

The screenshot shows the 'Definition' tab of a 'Formula parameter mapping' form. It contains the following fields:

- Data Mapping Type:** Insurance Product Premium Amount
- Master Entity:** FTOS_INSPA_Policy (indicated by a red dot and a dropdown arrow)
- Operation Name:** PA_Final_Premium
- Name:** PA_Final_Premium_1650531394377_IP (indicated by a red dot)

At the top right, there are two buttons: 'Save and close' and 'Save and reload'. At the top left, there is a back arrow icon. The form has three tabs: '1 Definition', '2 Input', and '3 Output', with '1 Definition' being the active tab.

4. Click **Save and reload**.
5. Go to the **Input** tab (second tab) and use the **Click to Map** buttons, next to each **formula input key** (in blue), in order to indicate the attribute (belonging to the previously chosen master entity) from which the system will extract the value needed for calculation.

Below, an example of an input mapping to the Policy entity:

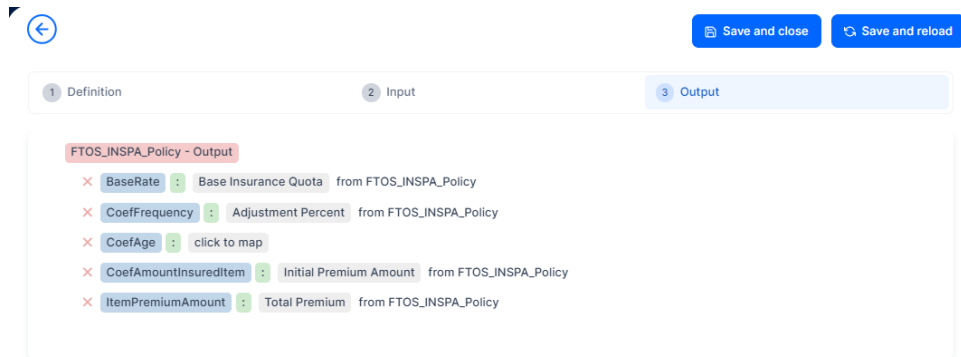
The screenshot shows the 'Input' tab of the 'Formula parameter mapping' form. The title is 'FTOS_INSPA_Policy - Input'. It lists four input keys with their corresponding attributes and sources:

- age:** Manual input (checked)
- coverage:** Policy Description from FTOS_INSPA_Policy
- frequency:** Payment Frequency from FTOS_INSPA_Policy
- suminsured:** Insured Amount from FTOS_INSPA_Policy

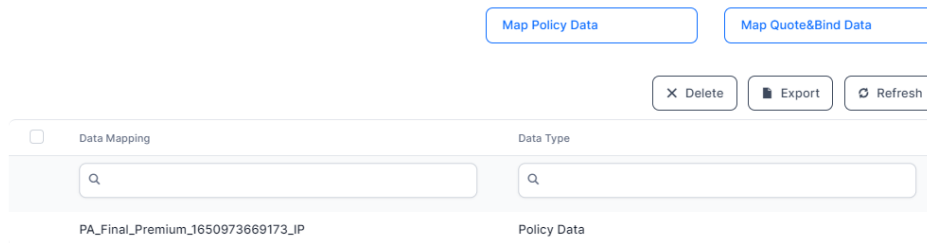
At the top right, there are two buttons: 'Save and close' and 'Save and reload'. At the top left, there is a back arrow icon. The form has three tabs: '1 Definition', '2 Input', and '3 Output', with '2 Input' being the active tab.

6. Click **Save and reload**.
7. Go to the **Output** tab (third tab) and use the **Click to Map** buttons, next to each **formula output key** (in blue), in order to indicate the attribute to which the system will make updates (insert the value obtained) after the calculation.

Below, an example of an output mapping for the Policy flow:



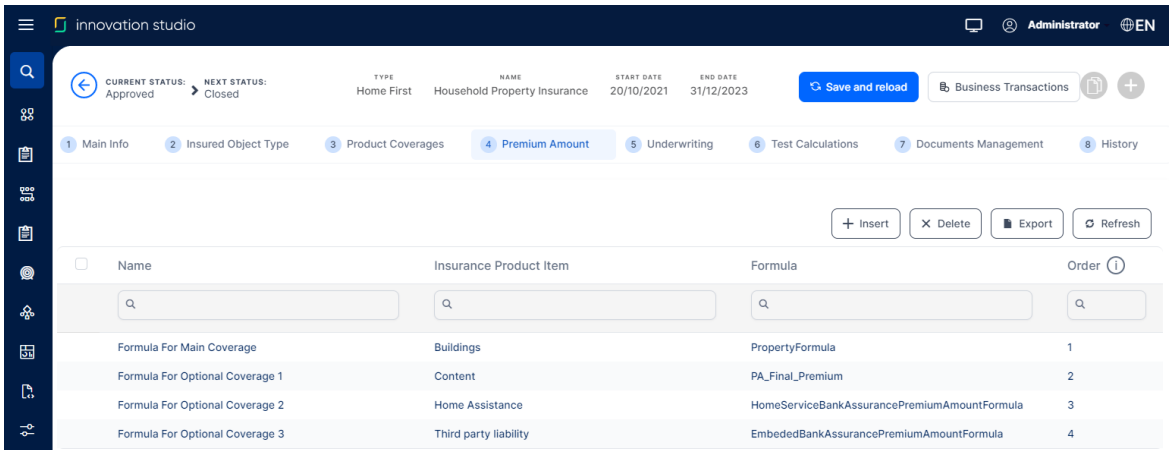
8. Once finished, click **Save and close**. You can now see the attached formula in the Data Mapping grid, at the bottom of the insert formula form, example below:



Add Insurance Coverage Formula

Prerequisite: You must have previously selected the **Per coverage** tariff type, in the **Main Info** product tab. This selection makes the **Add Insurance Product Item Formula** insert form available in the **Premium Amount** tab.

Below, an example of an insurance product with formulas attached to each of its coverages:

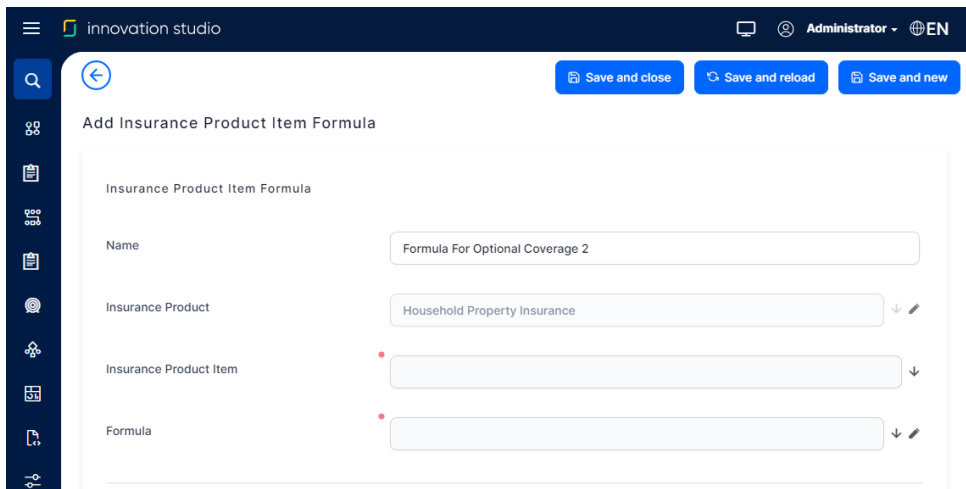


To add a formula that helps you calculate the premium amount per coverage (product item), follow the steps below:

1. Create A Configuration

1. In the **Premium Amount** tab, click **Insert**. The **Add Insurance Product Item Formula** insert form opens. Notice the name of the insurance product being auto-completed and not editable.

Below, an example of the form for adding insurance coverage formulas :



2. In the insert form, fill in a descriptive **Name** for the formula.
3. Use the dropdown to select the **Insurance Product Item** (coverage).

4. Use the picker (arrow) to select from the list a **Formula** to be appended to the coverage you indicated in the former step.
5. Once finished, click **Save and reload**.
6. Repeat these steps for all the coverages that your product has.

2. Map Data

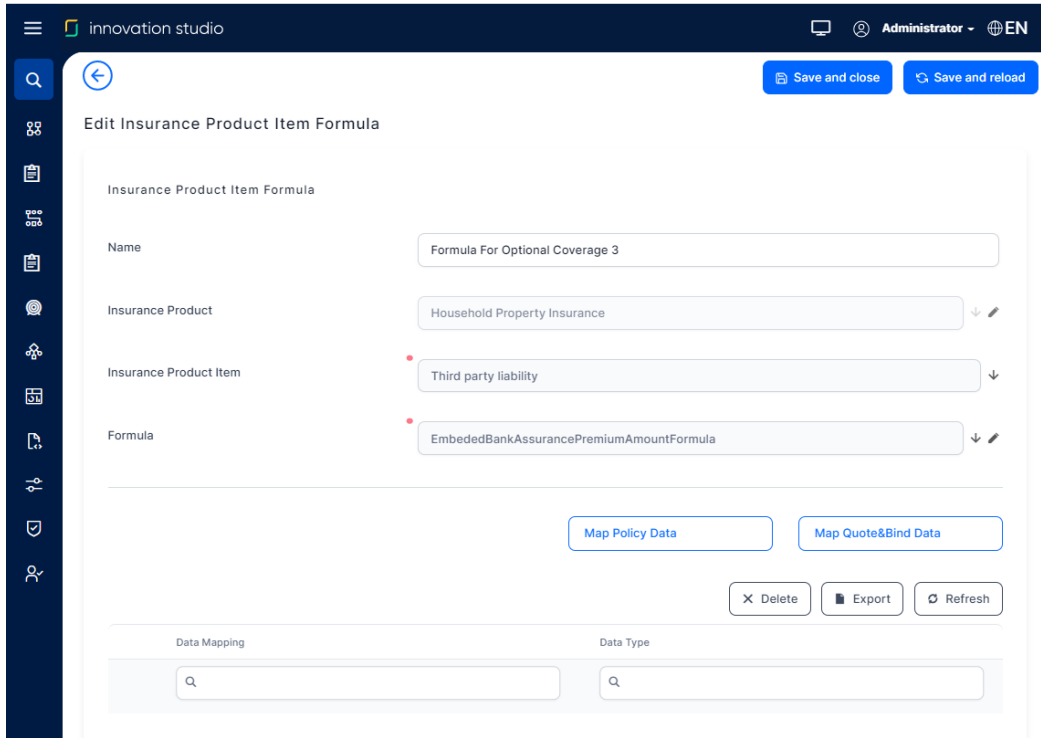
IMPORTANT!

Mapping data is a mandatory step in order for the system to know from where to fetch the product data necessary for the [formula engine](#) to run.

After **Save and reload**, the **Map Data** buttons become available.

These buttons allow you to define a single data mapping, for either a **Policy Admin** or a **Quote and Bind** flow. When clicking either of the buttons, you can select a **Master Entity** (any master entity that you need, for your calculations). This way, you get access to use the values of the selected entity's attributes as **formula input keys**, to obtain a certain output (and this output is registered on some other entity).

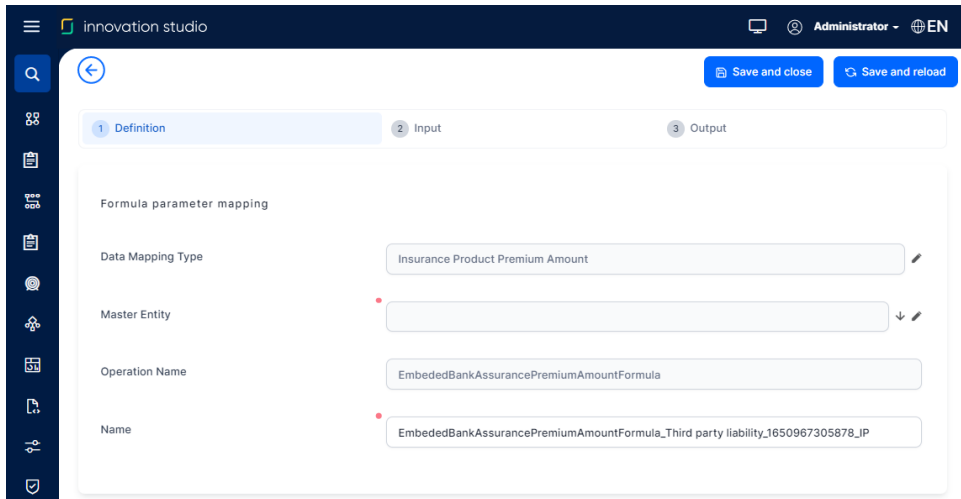
Below, an example of the insert form for the insurance coverage formula - with **Map Data** buttons available, after save and reload:



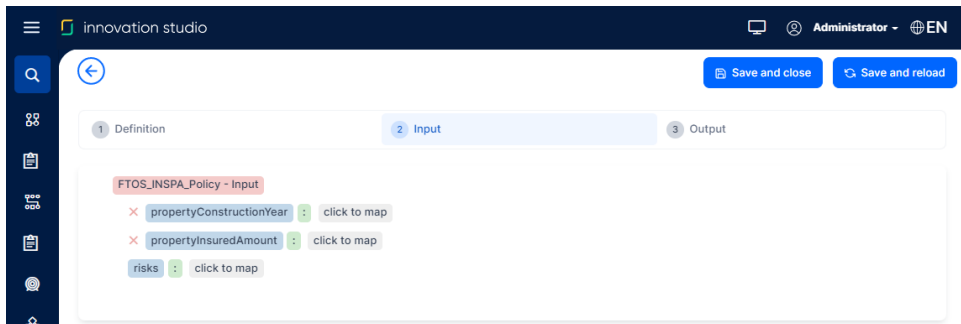
Proceed with mapping the necessary data for the attached formulas, as follows:

1. Click the **Map Data** button - either **Map Policy Data** or **Map Quote&Bind Data**, depending on the mapping you need to perform. The **Formula Parameter Mapping** form opens.
2. In the **Definition** tab (first tab), notice the **Data Mapping Type**, the **Operation Name** (the actual formula name), the **Name** (the mapping name) fields being auto-completed and not editable.
3. Use the **arrow selector** to indicate a **Master Entity** from which the system will extract the input keys for the formula.

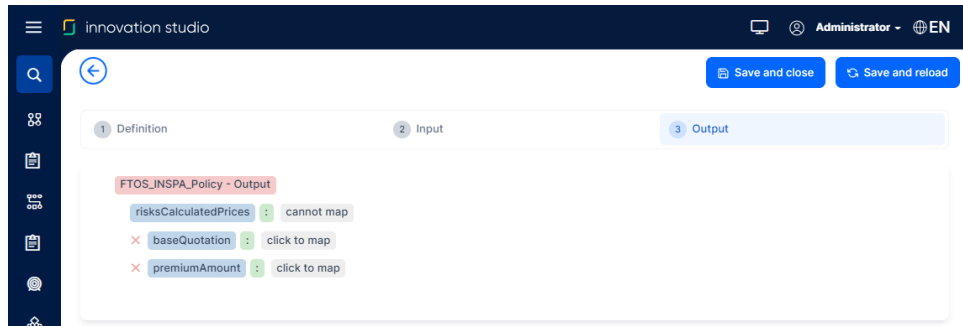
Below, an example of a **Formula Parameter Mapping** form, with definition details, for a per coverage data mapping:



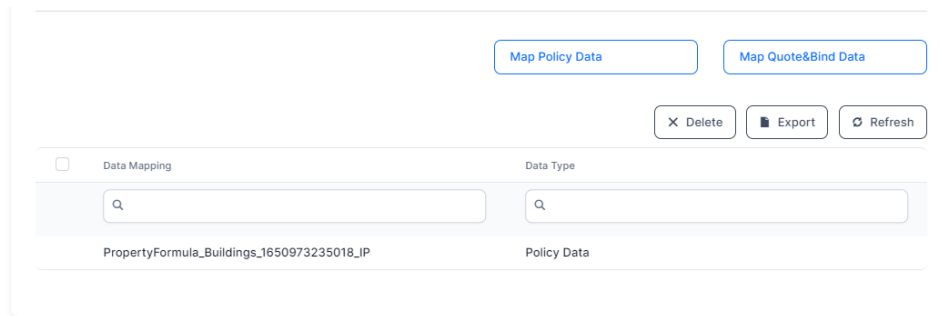
- Go to the **Input** tab (second tab) and use the **Click to Map** buttons, next to each **formula input key** (in blue), in order to indicate the attribute (belonging to the previously chosen master entity) from which the system will extract the value needed for calculation. Below, an example of an input mapping to the Policy entity:



- Click **Save and reload**.
- Go to the **Output** tab (third tab) and use the **Click to Map** buttons, next to each **formula output key** (in blue), in order to indicate the attribute to which the system will make updates (insert the value obtained) after the calculation. Below, an example of an output mapping to the Policy entity:



- Once finished, click **Save and close**. You can now see the attached formula in the Data Mapping grid, at the bottom of the insert formula form, example below:



NOTE

Changing the **Tariff Type** option set value, triggers the automatic removal of any formulas previously attached on a product, or product item (coverage).

You can design your formulas to calculate the **Premium Amount** for a **Product**, or a **Product Item** in steps, allowing for some input data to be received from third-party systems.

Check the Insurance Business Formulas page, for more formula use cases. For creating your own formulas, consult the [FintechOS Business Formulas](#) documentation. Additionally, the **Insurance Product Factory** solution comes with an optional import package containing some demo formulas. Read about them on the [Insurance Demo Formulas](#) page.

HINT

Continue to the [Underwriting](#) page.

Underwriting

The **Underwriting** tab allows you to configure how underwriting rules are applied to your insurance product, or product items (coverages), by attaching scoring formulas to them. You can also change previous configurations by deleting attached formulas and inserting new ones, if needed.

Based on the **Tariff Type** - previously chosen in the **Tariff Configuration** section, you can either see the **Insurance Product Underwriting** grid or the **Insurance Product Coverage Underwriting** grid (on this tab) .

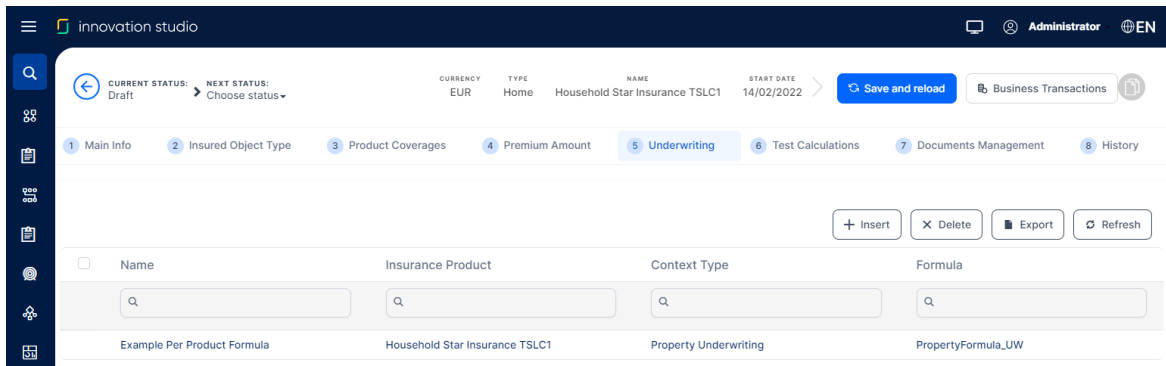
NOTE

Only one formula can be attached to a specified product, or product item (coverage), at a time.

Add Insurance Product Underwriting Formula

Prerequisite: You must have previously selected the **Per Product** underwriting type, in the **Main Info** product tab. This selection makes the **Add Insurance Formula Underwriting** insert form available in the **Underwriting** tab.

Below, an example of an insurance product with an attached insurance **Per Product** formula for underwriting:



To add a per product formula that helps you with scoring, follow the steps below:

1. Create A Configuration

1. In the **Underwriting** tab, click **Insert**.
2. The **Add Insurance Formula Underwriting** insert form opens. Notice the name of the insurance product being auto-completed and not editable.

Below, an example of an insurance product underwriting formula insert form, with some filled in details:

3. In the insert form, fill in a descriptive **Name** for the formula.
4. Use the picker (arrow) to select from the list a **Context Type** that will define the underwriting for your product - for example Property underwriting, Medical underwriting, General underwriting steps, and so on. Click **Ok**.

Below, an example of a **Context Type** list, with different list items:

Dialog box for adding context types. Buttons: Cancel, Remove, Insert, Ok. Search bar: [Q]. List items: Buildings Underwriting, Medical Underwriting, Property Underwriting, Underwriting.

NOTE
If you need, you can define (add) as many **Context Types** as necessary!

5. Use the picker (arrow) to select from the list a **Formula** to be appended to the current product, for underwriting purposes. Click **Ok**.
6. Click **Save and reload**.

Below, an example of an insurance product underwriting formula insert form, with all the details filled in:

Form titled "Add Insurance Formula Underwriting". Buttons: Save and close, Save and reload, Save and new. Fields: Name (Example Per Product Formula), Insurance Product (Household Star Insurance TSLC1), Context Type (Property Underwriting), Formula (PropertyFormula).

7. Move to the next step, below.

2. Map Data

IMPORTANT!

Mapping data is a mandatory step in order for the system to know from where to fetch the product data necessary for the **formula engine** to run.

After **Save and reload**, the **Map Data** buttons become available.

These buttons allow you to define a single data mapping, for either a **Policy Admin** or a **Quote and Bind** flow. When clicking either of the buttons, you can select a **Master Entity** (any master entity that you need, for your calculations). This way, you get access to use the values of the selected entity's attributes as **formula input keys**, to obtain a certain output (and this output is registered on some other entity).

Below, an example of the insert form for an insurance product underwriting formula - with **Map Data** buttons available, after save and reload:

← Save and close Save and reload

Edit Insurance Formula Underwriting

Insurance Formula Underwriting

Name Example Per Product Formula

Insurance Product Household Star Insurance TSLC1 ↓ ✎

Context Type Property Underwriting ↓ ✎

Formula PropertyFormula ↓

Map Policy Data Map Quote&Bind Data

✕ Delete Export Refresh

Data Mapping	Data Type
Q	Q

Proceed with mapping the necessary data for the attached formulas, as follows:

1. Click the **Map Data** button - either **Map Policy Data** or **Map Quote&Bind Data**, depending on the mapping you need to perform. The **Formula Parameter Mapping** form opens.
2. In the **Definition** tab (first tab), notice the **Data Mapping Type**, the **Operation Name** (the actual formula name), the **Name** (the mapping name) fields being auto-completed and not editable.
3. Use the **arrow selector** to indicate a **Master Entity** from which the system will extract the input keys for the formula.

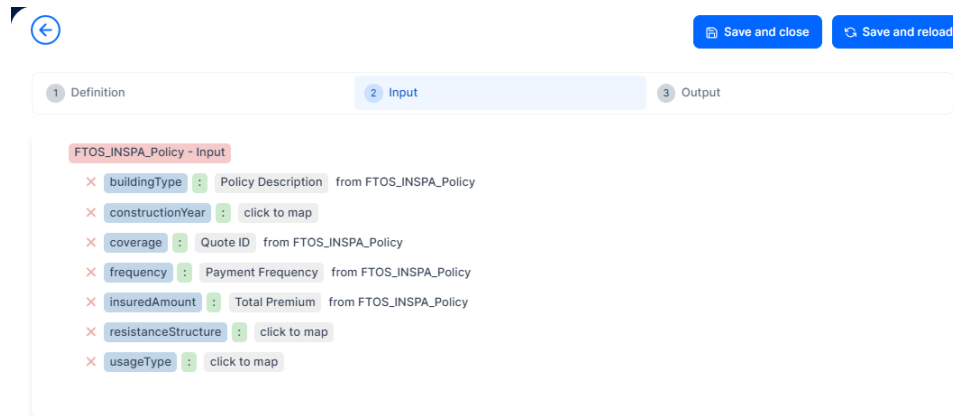
Below, an example of a **Formula Parameter Mapping** form, with definition details:

The screenshot shows a web form titled "Formula parameter mapping" with three tabs: "1 Definition", "2 Input", and "3 Output". The "Definition" tab is active. The form contains the following fields:

- Data Mapping Type:** Insurance Product Underwriting
- Master Entity:** A dropdown menu with a downward arrow icon.
- Operation Name:** PropertyFormula
- Name:** PropertyFormula_Property Underwriting_1650536331715_IPU

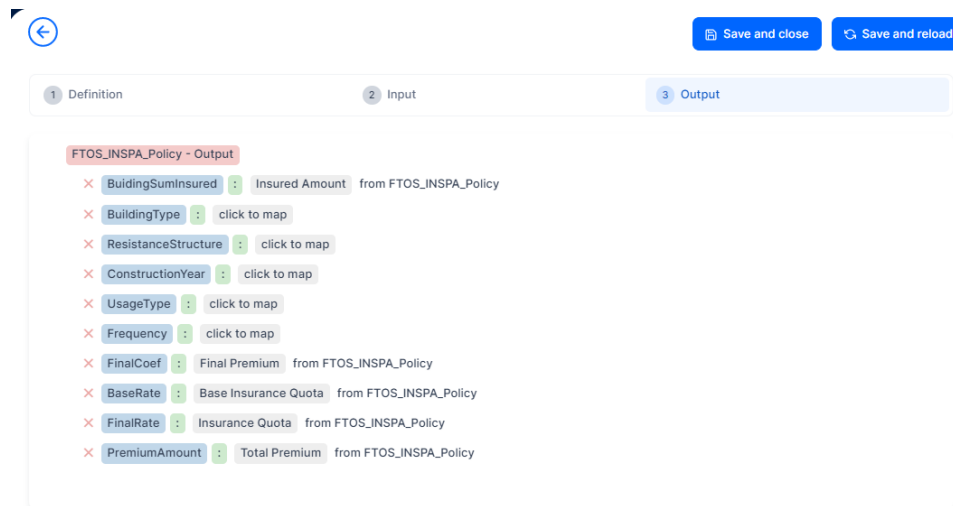
At the top right of the form, there are two buttons: "Save and close" and "Save and reload".

4. Go to the **Input** tab (second tab) and use the **Click to Map** buttons, next to each **formula input key** (in blue), in order to indicate the attribute (belonging to the previously chosen master entity) from which the system will extract the value needed for calculation. Below, an example of an input mapping to the Policy entity:

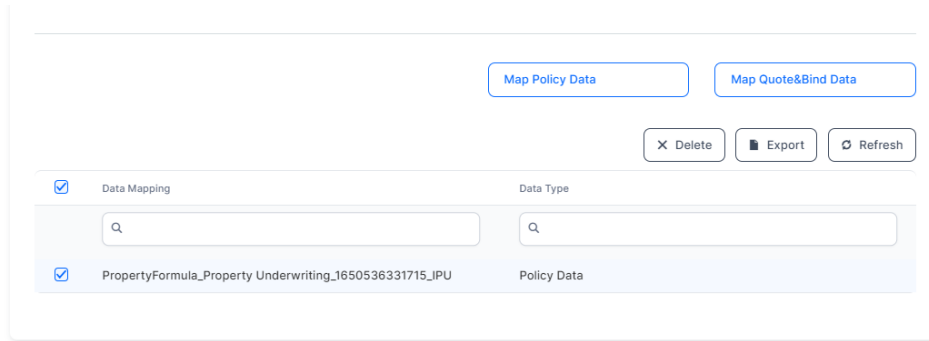


5. Click **Save and reload**.
6. Go to the **Output** tab (third tab) and use the **Click to Map** buttons, next to each **formula output key** (in blue), in order to indicate the attribute to which the system will make updates (insert the value obtained) after the calculation.

Below, an example of an output mapping to the Policy entity:



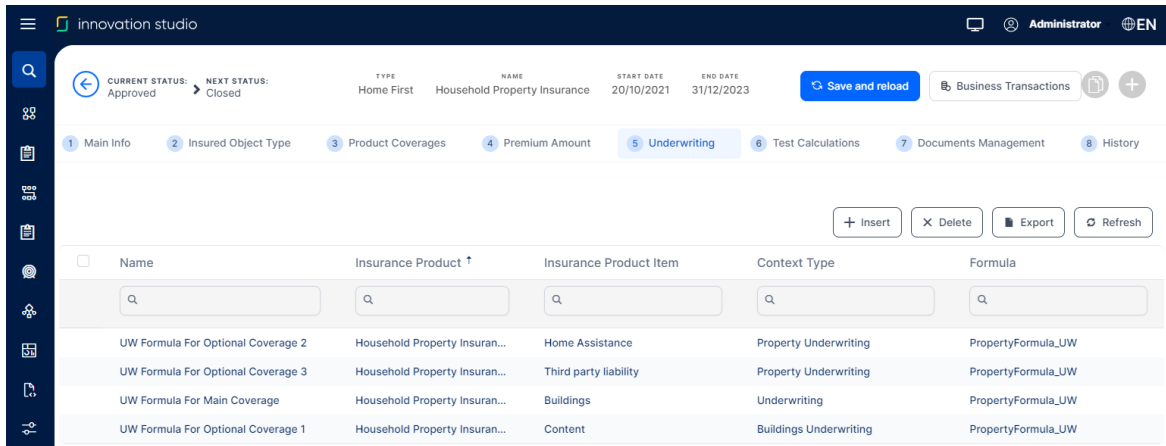
7. Once finished, click **Save and close**. You can now see the attached formula in the Data Mapping grid, at the bottom of the insert formula form, example below:



Add Insurance Product Coverage Underwriting Formula

Prerequisite: You must have previously selected the **Per Coverage** underwriting type, in the **Main Info** product tab. This selection makes the **Add Insurance Product Coverage Underwriting** insert form available in the **Underwriting** tab.

Below, an example of an insurance product with formulas for underwriting attached to its coverages (insurance product items):



To add a per coverage formula that helps you with scoring, follow the steps below:

1. Create A Configuration

1. In the **Underwriting** tab, click **Insert**.
2. The **Add Insurance Product Coverage Underwriting** insert form opens. Notice the name of the insurance product being auto-completed and not editable.

Below, an example of an insert form for coverage underwriting formulas, with some filled in details:

The screenshot shows a web application interface for 'innovation studio'. The main content area is titled 'Add Insurance Product Coverage Underwriting'. It contains a form with the following fields:

- Name:** A text input field.
- Insurance Product:** A dropdown menu with 'Household Property Insurance' selected.
- Insurance Product Item:** A dropdown menu.
- Context Type:** A dropdown menu.
- Formula:** A dropdown menu.

At the top right of the form, there are three buttons: 'Save and close', 'Save and reload', and 'Save and new'. The left sidebar contains various navigation icons.

3. In the insert form, fill in a descriptive **Name** for the formula.
4. Use the picker (arrow) to select from the list the **Coverage** (insurance product item) that needs underwriting.
5. Use the picker (arrow) to select from the list a **Context Type** that will define the underwriting for your coverage - for example Property underwriting, Medical underwriting, General underwriting steps, and so on.

Below, an example of a **Context Type** list, with different list items:

Dialog box for adding context types. Buttons: Cancel, Remove, Insert, Ok. Search bar: Name. List of context types: Buildings Underwriting, Medical Underwriting, Property Underwriting, Underwriting.

NOTE
If you need, you can define (add) as many **Context Types** as necessary!

- 6. Use the picker (arrow) to select from the list a **Formula** to be appended to the current coverage, for underwriting purposes.
- 7. Click **Save and reload**.

Below, an example of a coverage underwriting formula insert form, with all the details filled in:

Form titled "Add Insurance Product Coverage Underwriting" with fields: Name (UW Formula For Optional Coverage 3), Insurance Product (Household Property Insurance), Insurance Product Item (Third party liability), Context Type (Property Underwriting), Formula (PropertyFormula_UW). Buttons: Save and close, Save and reload, Save and new.

- 8. Move to the next step, below.

2. Map Data

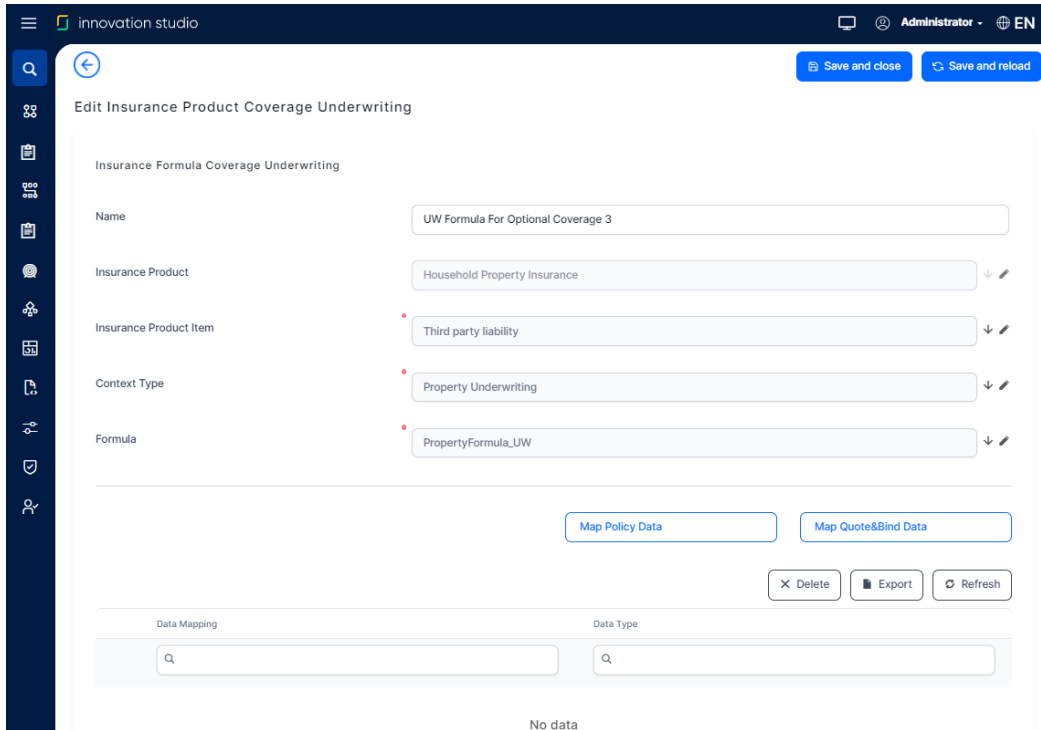
IMPORTANT!

Mapping data is a mandatory step in order for the system to know from where to fetch the product data necessary for the **formula engine** to run.

After **Save and reload**, the **Map Data** buttons become available.

These buttons allow you to define a single data mapping, for either a **Policy Admin** or a **Quote and Bind** flow. When clicking either of the buttons, you can select a **Master Entity** (any master entity that you need, for your calculations). This way, you get access to use the values of the selected entity's attributes as **formula input keys**, to obtain a certain output (and this output is registered on some other entity).

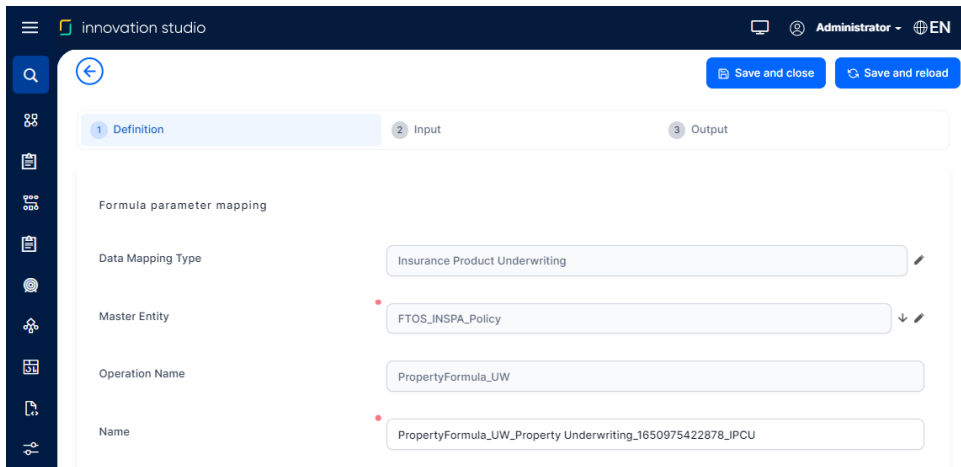
Below, an example of the insert form for a coverage UW formula - with **Map Data** buttons available, after save and reload:



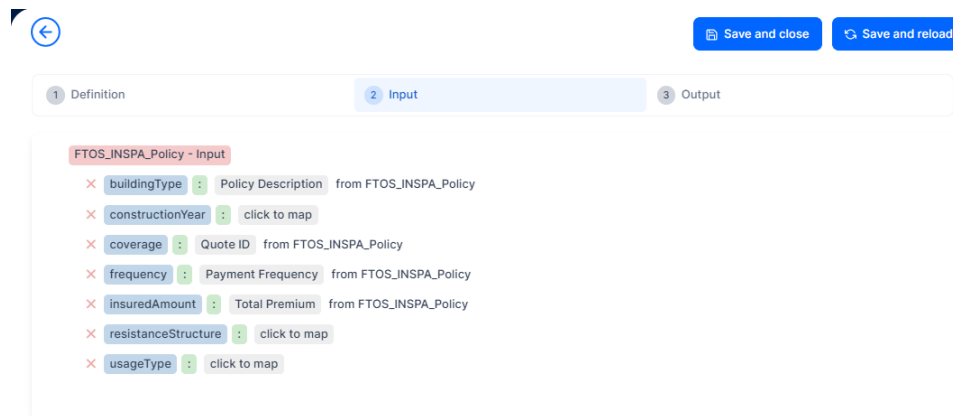
Proceed with mapping the necessary data for the attached formulas, as follows:

1. Click the **Map Data** button - either **Map Policy Data** or **Map Quote&Bind Data**, depending on the mapping you need to perform. The **Formula Parameter Mapping** form opens.
2. In the **Definition** tab (first tab), notice the **Data Mapping Type**, the **Operation Name** (the actual formula name), the **Name** (the mapping name) fields being auto-completed and not editable.
3. Use the **arrow selector** to indicate a **Master Entity** from which the system will extract the input keys for the formula.

Below, an example of a **Formula Parameter Mapping** form, with definition details:

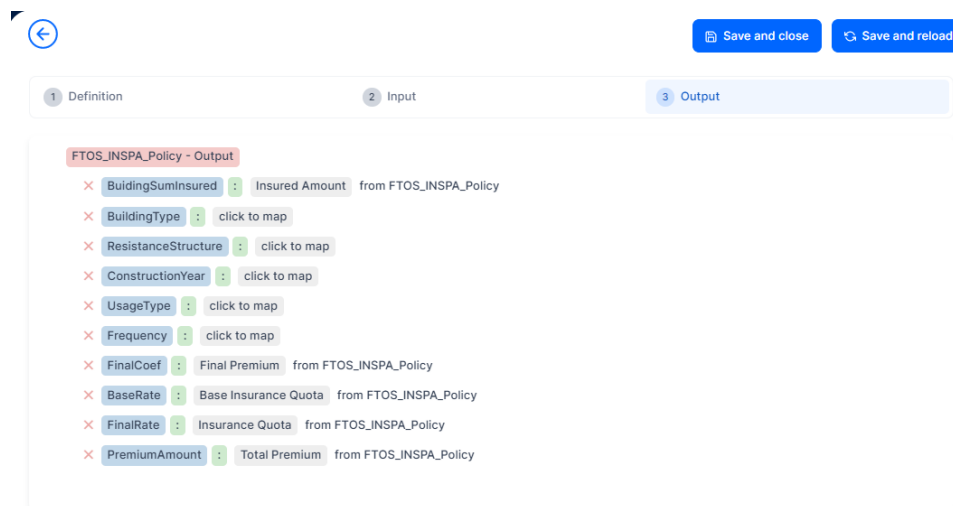


4. Go to the **Input** tab (second tab) and use the **Click to Map** buttons, next to each **formula input key** (in blue), in order to indicate the attribute (belonging to the previously chosen master entity) from which the system will extract the value needed for calculation. Below, an example of an input mapping to the Policy entity:

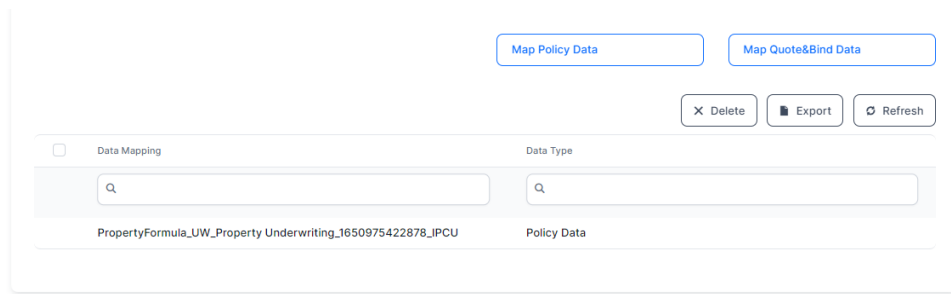


5. Click **Save and reload**.
6. Go to the **Output** tab (third tab) and use the **Click to Map** buttons, next to each **formula output key** (in blue), in order to indicate the attribute to which the system will make updates (insert the value obtained) after the calculation.

Below, an example of an output mapping to the Policy entity:



7. Once finished, click **Save and close**. You can now see the attached formula in the Data Mapping grid, at the bottom of the insert formula form, example below:



NOTE

Changing the **Underwriting Type** option set value, triggers the automatic removal of any formulas previously attached on a product, or coverage.

You can design your formulas to calculate the **Underwriting** result for a **Product**, or a **Product Item** in steps, allowing for some input data to be received from third-party systems.

Check the Insurance Business Formulas page, for more formula use cases. For creating your own formulas, consult the [FintechOS Business Formulas](#) documentation. Additionally, the **Insurance Product Factory** solution comes with an optional import package containing some demo formulas. Read about them on the [Insurance Demo Formulas](#) page.

HINT

Continue to the [Test Calculations](#) tab.

Test Calculations

With **FintechOS** technology, you can [create your own insurance formulas](#), process data for modeling and interpretation, and cast advanced pricing models. The [Business Formulas](#) solution, for example, allows you to design formulas that can be attached to different calculating targets (e.g. product pricing/ steps in underwriting flows), and, by

doing so, reduce the completion time for those calculations. Moreover, the solution has also a testing feature that allows you to test the formulas you design, before activating them.

Every insurer needs to calculate, calculate and recalculate how the variables related to the insured object influence say the cost of each peril (or a group of perils) and use the resulting knowledge when designing their products. This is where the **Insurance Product Factory** comes into play: the solution helps you with shortening the time from formula to product design, since it provides access to your formulas and a testing functionality **without leaving the context of your product**. Subsequently, the **Test Calculations** functionality helps you to test your formulas, thus contributing to fine-tuning and improving the general performance of your product.

The **Test Calculations** tab allows you to:

- perform the calculations according to the **Tariff Type** and **Underwriting Type**, set for the specified per product, or product item (coverage).
- test the pricing and underwriting formulas attached to your product, or coverages.
- see the results you get with different **Testing Scenarios**.
- save the tests you consider relevant, for further processing.
- perform and store unlimited numbers of tests.

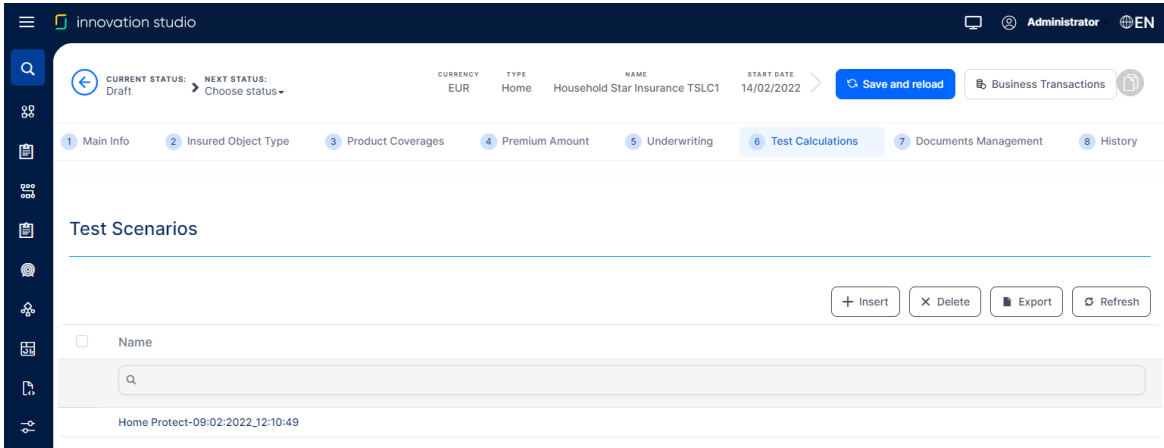
NOTE

Only one formula can be attached to a specified product, or coverage, at a time.

Test Calculations View

The **Test Calculations** tab offers you an overview of all the test scenarios registered for the current product. This is an all-inclusive view; yet, you can also use the **Search** option in order to find a specific peril record.

Below, an example of the **Test Calculations** main grid, with some registered scenarios:



Inside this grid,

- To inspect a record from the grid, double-click it.
- To add a new test calculations record, click **Insert**, at the top of the grid.
- To edit a record from the grid, double-click it and make your updates. Next, click **Save and close**.
- To delete a record from the grid, select it and click **Delete**, at the top of the grid.
- To export a record from the grid, select it and click **Export**, at the top of the grid.

Create A Test Scenario

To add a test scenario that helps you test a formula attached to a product, or product item (coverage), follow the steps below:

1. In the **Test Calculations** tab, click **Insert**. The **Add Test Scenario** insert form opens.
The form displays the fields according to the previously chosen **Tariff Type** - namely, you can create a test calculation scenario either for an **Insurance Product Formula** or for an **Insurance Product Item Formula**, depending on the product configurations.
2. From the dropdown list select a **Scenario Type** to be used for calculation. The options are: **Premium Amount** calculation or **Verify Underwriting**.

3. Use the picker (arrow) to select the **Formula** to be used for calculation:
 - For the **Premium Amount** scenario type, the picker list contains [only the premium calculation formulas that are mapped to your product, or coverages](#).
 - For the **Verify Underwriting** scenario type, the picker list contains [only the underwriting formulas that are mapped to your product, or product items](#).
4. Use the option set to set a **Testing Type**. The available options are: **Individual testing** and **Batch testing**.
5. Click **Save and reload**. Continue to **Individual testing** or **Batch testing**, below, depending on your testing scenario type.

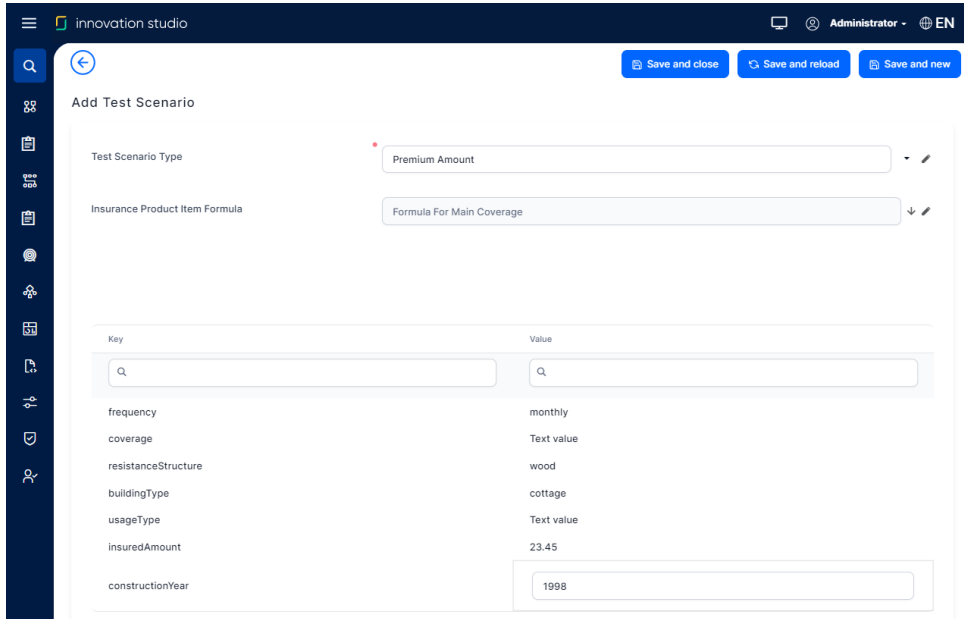
Below, an example of the test scenario insert form:

Individual testing

After **Save and reload**, the **Key & Value** grid becomes available.

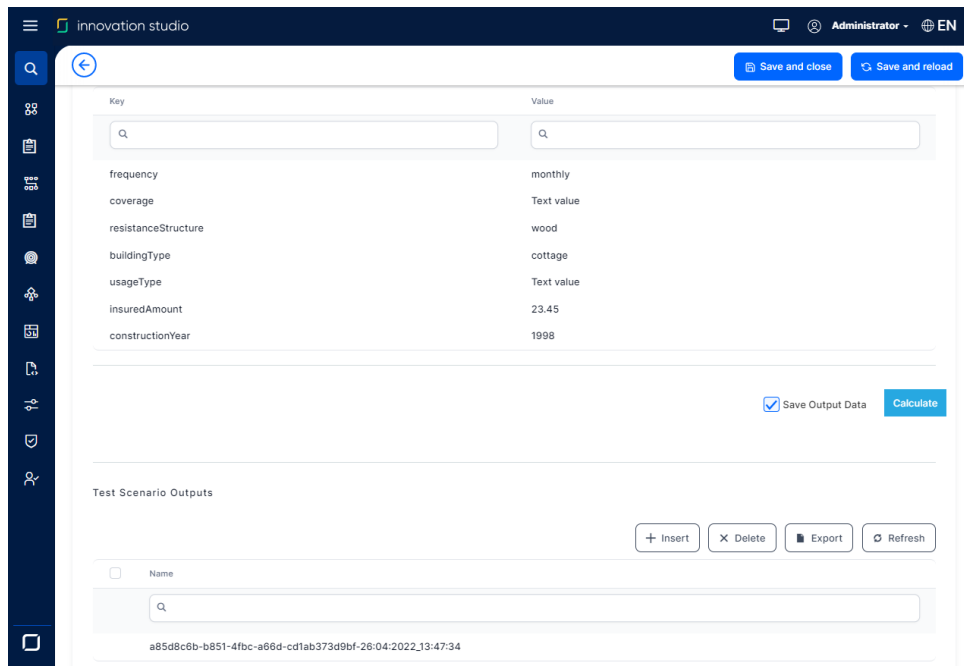
1. Inside this grid, fill in the values for the exposed keys. Pay attention to that fact that the text values are case sensitive!

Below, an example of the insurance test scenario insert form, with some filled in details in the **Key & Value** grid:



2. Click **Save and reload**. The **Calculate** button and the option to (check) **Save the Output Data** become available. The **Test Scenario Outputs** grid becomes available, also, at the bottom of the form.

Below, an example of the insurance test scenario insert form, with the Calculate and Save the Output Data buttons, available after **Save and reload**:



3. Press the **Calculate** button and see the results of your test displayed as a secondary **Key & Value** grid, right under the first one. If you selected the **Save the Output Data** option, you can notice your test results being logged on, in the **Test Scenario Outputs** grid, at the bottom of the form.
4. You can repeat the steps 6, 7, and 8 for as many test calculations you need to perform, for this particular scenario.
5. Click **Save and close**.

Batch testing

After **Save and reload**, the following buttons become available: **Batch file template**, **Upload**, **Run** and **Download results**.

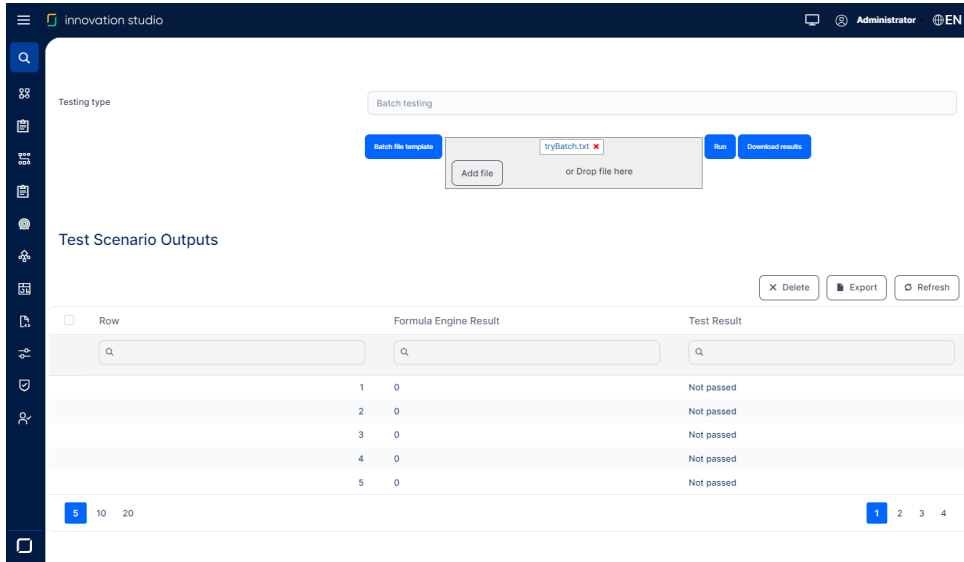
1. Click the **Batch file template** button to download the template for data processing. The template is an excel file that has the following header **structure**:
Number; **Keys** (the keys defined in the formula); **Expected result**.

NOTE

The keys are loaded into the template in accordance with the formula you previously indicated for test scenario. For example, if you use a property formula, you might see the **resistanceStructure**, **sumInsured** and **frequency** keys.

2. Fill in the template with your testing data.
3. Upload the file using the **Add File** button.
4. After the upload, press **Run**.
5. The system tests each row of the template (identified by the **Number** field) and compares the expected result vs the actual result of the formula test. The result of the test is passed if the result of the formula test is equal to the expected result.
6. The system displays a new grid that shows each line from the uploaded file, with the following information: **Row**, **Formula Engine Result** and **Test Result** (Passed/ Not passed). You can click on each line to see the results for that particular test item, displayed in read-only format.
7. Click **Download results**, to download the file containing the results shown in the test calculation grid.

Below, an example of the Batch testing feature, with the available buttons and some displayed output values, resulted after a batch testing run, for a property product:



HINT

Continue to the [Documents Management](#) tab.

Documents Management

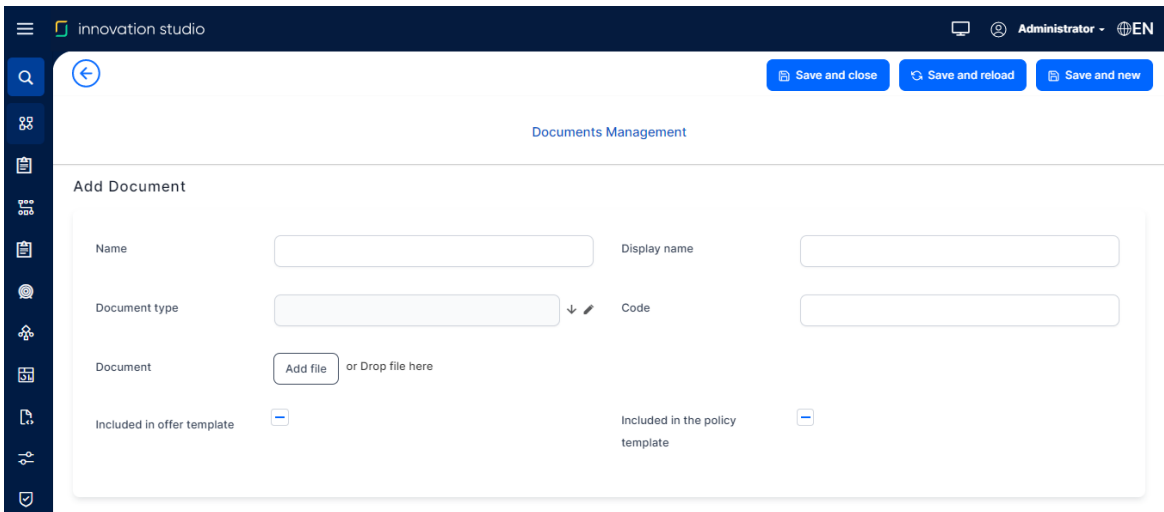
This section allows you to manage the documents that are associated with your product. For example:

- terms and conditions for the current insurance product,
- the mandatory general presentation of the product as requested by the authorities,
- mandatory clauses to be included in contracts based on different criteria,
- sample presentations that are to be used in the quote & apply journey or for different customer personas, and more.

For each **Document** that you upload, the following information must be provided:

Field	Description
Name	The name of your document.
Display name	The display name of your document.
Document type	From the dropdown, choose between Policy, Terms & Conditions or IPID - Insurance Product Information Document - which is a type of document necessary for the Quote & Buy journeys.
Code	The Code for your document.
Included in offer template	Check the box if the document must be included in offer template.
Included in the policy template	Check the box if the document must be included in the policy template.

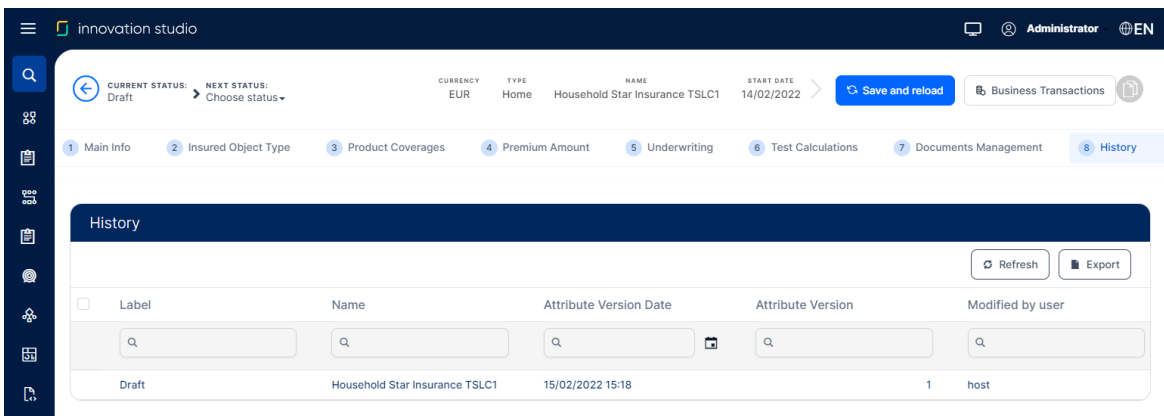
Click **Add File** to upload your document and then click **Save and reload** to continue your journey.



HINT
Continue to the [History](#) tab.

History

A history of the product allows you to understand how your product evolved over time and decide on updating its course, eventually. You go to the **History** tab when you need to have an overview over the product's life cycle, inform yourself about previous product versions (approved or unapproved), their workflow status, or to obtain details about users who modified the product.



HINT

Product **status** determines if your insurance product is used in digital journeys targeting potential customers. In order to make it available to different digital journeys, the product should be in the **Approved** status. For more details, see the [Product Life Cycle](#) page.

Managing the Product Factory

Feature Overview

The **Insurance Product Factory** makes it easier for you to streamline the creation, deployment and maintenance of insurance products, and product sub-components. This direct coupling along the product lifecycle makes place for better performance, also when it comes to portfolio management.

The products that you develop might transition between different business statuses - like **Draft, Approved, Closed**, or they might be updated to a new version, they might be the subject of special offers, or they might be part of recurring offers (but with limited availability), and so on. For all these cases and more, you use **Insurance Product Factory** to be on track with accurately recording and housing the said changes, control product behavior and, also, to efficiently organize the updated information.

Maintaining the products while they are active (exposed through quotation flows) and inquired into by multiple API calls, is yet another superlative of the solution.

IMPORTANT!

For specific details about creating a product, consult the [Creating Insurance Products](#) page.

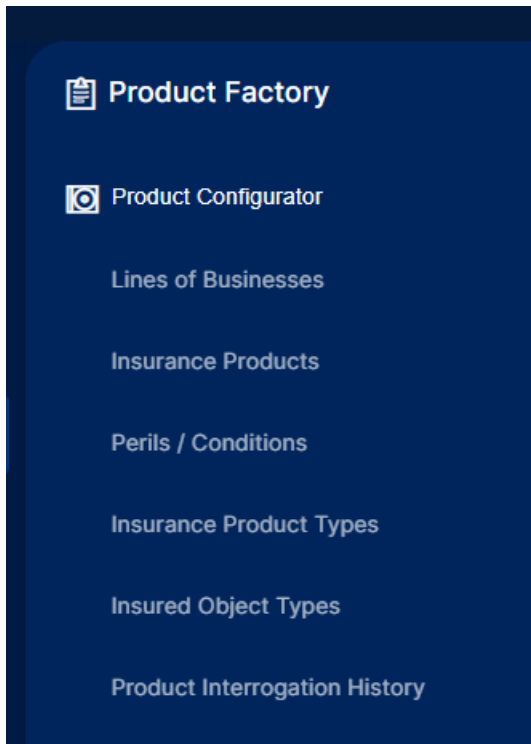
Related to this topic:

- [Insurance Lines Of Businesses](#) - details about creating and maintaining LOB classifications.
- [Insurance Products](#) - general details about insurance products.

- [Perils And Conditions](#) - details about how to create and configure perils, conditions, or both.
- [Insurance Types](#) - details about how to create and configure types.
- [Insured Object Types](#) - details about how to create and configure object types.
- [Product Interrogation History](#) - details about how to access the API interrogation history of a product.
- [Insurance Business Formulas](#) - see below details about how to attach your insurance formulas to your products.

Insurance Product Factory Menu Items

Below, an example of the **Insurance Product Factory** menu items:



HINT

The **Insurance Product Factory** maintains the underlying data in a consistent data model. Consequently, you have a reliable data model for your insurance products that you can reference when you build your digital journeys. This allows you to

manage your product portfolio at will, without having to re-code your digital journeys every time an insurance product is added, updated, or retired.

Insurance Business Formulas

Insurance companies monitor the performance of their products to ensure that they can generate the required level of profit, which is the difference between net premium collected (income) and the cost of claims paid (expenses). Consequently, insurers must constantly adjust premium rates and coverage to maintain a balance between profitability and price competitiveness. Nowadays, this task becomes increasingly complex and requires an ever increasing amount of calculations.

Introducing the **FintechOS Business Formulas...** a solution that helps you reduce completion time for different types of calculations. It allows you to **create, test, and activate** your own insurance formulas (see the use cases below). You can also **version** your formulas.

In short, you configure the formula input - an input argument, or a list of **arguments**, a data set (which can contain financial or non-financial **data**), then a formula expression. Next, you test the formula. Once finished, you activate the formula. Once active, the formula can be mapped to different calculation targets - such as a product, or a step in a customized underwriting process. Then, every time the formula is called, the **Formula Engine** handles all the various calculations for you and generates the desired outputs, in an accurate manner.

However, a rate is only valid for a set time period... Underwriters set the rates (pricing level) for a risk based on the underlying frequency and cost of claims, and other variables that might change their values. Sometimes rates are too cheap (don't cover claim costs) or too expensive (uncompetitive) so underwriters must replace them with an amended rate which will be charged from an **effective date**. Specifically for this case, formula **versioning** allows you to ensure that the correct rating is used for charging, for any risk.

Hence, with **Formulas**, the complexity of a granular pricing engine is decoupled from the digital quotation system, or at least diminished, and kept in a dedicated environment. This helps control the cost of development and reduces the maintenance and update effort for your portfolio of insurance products.

Insurance Formulas Use Cases

FintechOS Formulas can support various business needs. But here are some examples that might interest insurers:

- create formulas for premium amount calculations, insurance peril scoring, and underwriting evaluations.
- create formulas for split calculations - calculations applied per product items (coverages), instead of the whole product, or calculations configured differently for individual or group purchases.
- create calculation (or scoring) scenarios - series of formulas, executed in steps, following a defined order index.
- create testing scenarios for your formulas - such as testing different product prices.
- create formulas for underwriting insurance perils for different products, or product items (coverages).
- create formulas for determining whether the policy proceeds further with an automatic flow or if it needs to go on a manual approval flow.
- create formulas (with steps) that produce a status or decision regarding a quote - for example, establishing whether the proposed perils are insurable, after receiving data from a third-party system.
- calling **formulas** from a quote configuration digital journey, thus allowing customers to configure their own insurance quote in real time.

HINT

Formulas are vital for making your **Insurance Product Factory** work at its full capacity.

Bring Your Formulas To The Factory

The **Insurance Product Factory** solution provides access to your formulas and a testing functionality without leaving the context of your product, thus shortening the time from rating calculations to product launch.

In **Innovation Studio**, inside every product record:

- The **Premium Amount** tab allows you to attach **premium calculation formulas** to the selected product, or its coverages (product items). For details, see the [Premium Amount](#) page.
- The **Underwriting** tab allows you to attach **scoring/ underwriting formulas** to the selected product, or its coverages (product items). For details, see the [Underwriting](#) page.
- The **Test Calculations** tab allows you to **test your formulas** based on different scenarios. For details, see the [Test Calculations](#) page.

NOTE

Only one formula can be attached to a specified product, or coverage, at a time.

Check the following pages to find out about:

- [Attaching Formulas](#) - for technical details about how formulas and products work together.
- [Defining Premium Splits](#) - for technical details about how the split per coverage feature works.
- [Mapping Formulas](#) - for technical details about how the data mapping works.
- [Testing Formulas](#) - for technical details about how the formulas testing works.
- [Demo Formulas](#) - for the demo formulas description.

Demo Formulas

The **Insurance Product Factory** solution includes also an optional digital asset containing some demo formulas, so that you don't start the creation of your insurance formulas from scratch.

The following formulas are included:

- Bankassurance_UW_Formula_Final
- EmbeddedBankAssurancePremiumAmountFormula
- HomeServiceBankAssurancePremiumAmountFormula
- PA_Final_Premium
- PropertyFormula
- PropertyFormula_UW

You can read about them in the [Demo Formulas](#) page. A short description of the **Import Formulas** digital asset can be found in the [Digital Assets](#) page.

HINT

For more details about configuring formulas, see the [Business Formulas](#) documentation and this [Create Formula For Risk Scoring](#) tutorial.

Insurance Lines Of Businesses

In many countries, insurers must register their **Lines of Businesses** (LOBs) with their insurance supervisory authority and make insurance offers according to their authorized LOBs. A **Line Of Business** (LOB) is a general classification of business used by the insurance industry. It has a regulatory and accounting definition - such as **Fire, Motor, Personal Accident**, or **General Third Party Liability**, and meets a rather rigidly defined set of insurance policies. Consequently, insurers cannot establish policies outside the scope of their registered LOB. Besides that, some insurance companies might have multiple authorized LOBs, depending on how many types of insurance they want to sell.

This is where the **Insurance Product Factory** comes into play: the solution has an inbuilt feature, which allows you to create and manage classification items, and hierarchies. Once defined, a classification can be attached to a product, and then the policies based on that product further inherit the same classification.

The **Lines Of Businesses** functionality allows you to build your own **LOB** nomenclature, according to your activity, with the following levels of granularity:

- **Class of Business** - Eg.: Property, Casualty, Life, Health;
- **Category of Business** - Eg.: Personal Lines, Commercial Lines;
- **Line of Business** - Eg.: Building, Content, Motor, Travel, Health;
- **Line of Business Subtype** - Eg.: Individual Health.

IMPORTANT!

The solution does not allow you to delete any of the LOB records (class, category, line or subtype) from the grid.

To configure your **Lines Of Business**, follow the below steps:

1. In **Innovation Studio**, at the top left corner of your screen, click the main menu icon to open the main dropdown list.
2. From this main list, click **Insurance Product Factory**. A second panel opens, to the left.
3. Next, click **Lines Of Businesses** to go to the **Lines Of Businesses** page.

On the **Lines Of Businesses** page:

- To inspect a record from the grid, double-click it.
- To add a new record, click **Insert**, inside the available sections:

Class of Business

Use the **Insert** button to insert a new record on the **Class of Business** grid.

Below, an example of this grid:

Lines of Businesses

Class of Business	
<input type="button" value="+ Insert"/> <input type="button" value="Refresh"/> <input type="button" value="Export"/>	
<input type="checkbox"/>	Class of Business
<input type="text" value="Q"/>	
	Life
	Aviation
	Casualty
	Health
	Property

Category of Business

Use the **Insert** button to insert a new record on the **Category of Business** grid.

Below, an example of this grid:

Category of Business	
<input type="button" value="+ Insert"/> <input type="button" value="Refresh"/> <input type="button" value="Export"/>	
<input type="checkbox"/>	Category of Business
	Class of Business
<input type="text" value="Q"/>	<input type="text" value="Q"/>
Renewable Term Life Insurance	Life
Term Life - Individual	Life
Term Life Individual - for Group Cover	Life
Aircraft Crew Personal Accident Cover	Aviation
Aircraft Hull and Aircraft Liability	Aviation
<input type="button" value="5"/> 10 20	1 <input type="button" value="2"/> 3 4 5

Line of Business (LOB)

Use the **Insert** button to insert a new record on the **Line of Business (LOB)** grid.

Below, an example of this grid:

Line of Business (LOB)

<input type="checkbox"/>	Line of Business	Category of Business	Class of Business
	<input type="text" value="Q"/>	<input type="text" value="Q"/>	<input type="text" value="Q"/>
	Term Life for Employees	Term Life Individual - for Gro...	Life
	Public Liability for Bodily Injury	Public Liability	Casualty
	Disability	Personal Accident	Health
	Fixed Property - Content	Fixed Property	Property
	Boat Insurance	Motor	Property

10 20
 1 3

Line of Business Subtype

Use the **Insert** button to insert a new record on the **Line of Business Subtype** grid.

Below, an example of this grid:

Line of Business Subtype

<input type="checkbox"/>	LOB Subtype ↓	Line of Business	Category of Business	Class of Business
	<input type="text" value="Q"/>	<input type="text" value="Q"/>	<input type="text" value="Q"/>	<input type="text" value="Q"/>
	Fixed Property Content - Commercial	Fixed Property - Content	Fixed Property	Property
	Convertible - Whole Life (individual cover)	Convertible Term Life Insurance	Convertible Term Life Insurance	Life
	Convertible - Variable Universal (individual cover)	Convertible Term Life Insurance	Convertible Term Life Insurance	Life
	Convertible - Variable Universal (grup cover)	Convertible Term Life Insurance	Convertible Term Life Insurance	Life
	Convertible - Universal Life (individual cover)	Convertible Term Life Insurance	Convertible Term Life Insurance	Life

10 20
 1 2 3 4 6 7

HINT

You can export one or more records by pressing **Export**, inside the available sections.

Insurance Products

The **Insurance Product Factory** enables you to create and manage a great variety of insurance products. Depending on your customers needs, you might want to create very simple products - containing the [insurable object](#) and one base (main) coverage. Yet, for another target market, you might want to design a more sophisticated product - containing one insurable object, two [main coverages](#) and more optional coverages (riders). Besides that, you might need to develop different products in line with your authorized different [Lines of Businesses](#) (LOBs).

The Insurance Products functionality provides access to your insurance business formulas and a testing functionality **without leaving the context of your product**, thus helping you focus and shortening the time from rating calculations to product launch. Additionally, the solution makes top down, as well as bottom up, rating models applicable when building your products. This translates to the calculations for the [premium amount](#) and [underwriting](#) rules being available for each product coverage, and also for the whole product. Depending on your model, you can add constraints as underwriting steps, in order to reach only the intended audience.

Once activated, a product is [digital journey-ready](#) and you can expose it to potential customers through different digital channels. Cloning your products, helps you replicate successful products to different segments/ geographies, **with a click**. Versioning products, lets you amend, or enrich, products that are already live.

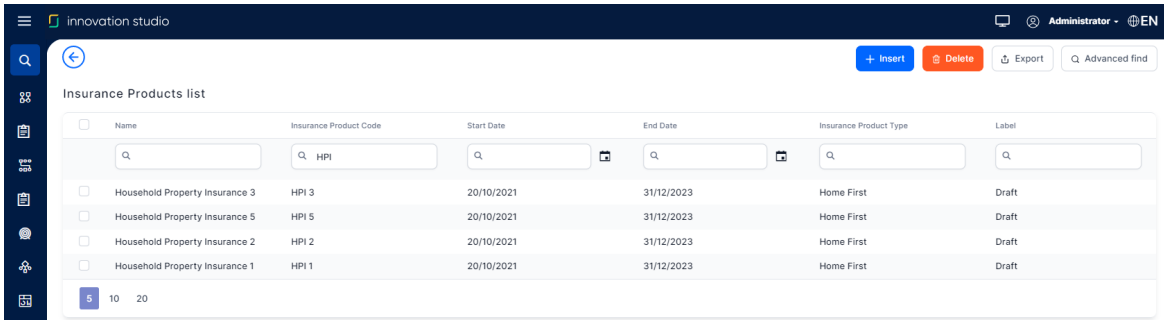
Since building or maintaining a product is rarely the work of a single individual, the solution helps you with tracking changes, also. It logs any modifications by user roles, including those done automatically by the system - such as transitioning the product to a **Closed** state, if an **availability time limit** was set on the product.

IMPORTANT!
FOR NEW PRODUCTS: start your creation journey from the [Creating Insurance Products](#) page.

Insurance Products View

In **Innovation Studio**, in the **Insurance Products** section, you have an overview of all the product records registered in your system - your portfolio of **Insurance Products**. This is an all-inclusive view; yet, you can also use the **Search by** option (in the list header) in order to find a certain record. For example, for easier processing, if you want to view all the products in **Draft** status, you can use the **Search by Label** option, and sort all your products accordingly.

Below, you can see an example of a list view for products and a product search example:



Follow these steps to view your **Insurance Products**:

1. In **Innovation Studio**, at the top left corner of your screen, click the main menu icon to open the main dropdown list.
2. From this main list, click **Insurance Product Factory**. A second panel opens, to the left.
3. Next, click **Insurance Products** to go to the **Insurance Products list**.

On the **Insurance Products list** page:

- To inspect a record from the grid, double-click it.
- To **add a new product** record, click **Insert**, at the top right corner of the page. For detailed instructions, go to the [Creating Insurance Products](#) page. For alternative ways to add product records, see also the [Product Operations](#) page.
- To edit the record of a **Draft** product, double-click it and press **Edit**.
- To edit the record of an **Active** product, click the versioning (+) button, at the top right corner of the page. For more details about versioning, see the [Product Operations](#) page.
- To delete a record from the grid, select it and click **Delete**, at the top right corner of the page.
- To export records, press **Export**, at the top right corner of your screen.

HINT

For details about available product operations consult the [Product Operations](#) page.

Product Operations

The following operations are available for interacting with your products portfolio, in **Innovation Studio**:

Creating Draft Insurance Products

There are times when you need to create a draft product, rapidly. While the product is in **Draft** business status, the form is still editable and you can complete the product creation journey at a later time.

To create a draft product, please follow the next steps:

1. In **Innovation Studio**, at the top left corner of your screen, click the main menu icon to open the main dropdown list.
2. From this main list, click **Insurance Product Factory**. A second panel opens, to the left.
3. Next, click **Insurance Products** to go to the **Insurance Products List**.

4. On the **Insurance Products List** page, click **Insert**, at the top right corner of the page.
5. The **Insurance Product** form opens. Fill in only the mandatory fields, and you will be able to save the record as a **Draft**.
6. Click **Save and close**.

The following are the mandatory fields, that you must fill in before **Save and close**:

Field	Description
Insurance Product Type	Click the dropdown to select a Type of insurance for your insurance product - ex. Auto, Health, Home, Travel. See Insurance Types for details.
Insurance Product Code	Fill in the code of your insurance product.
Name	Insert the name of your insurance product.
Currency	From the dropdown, select the currency for your insurance product.
Prorata Type Configuration	Set the proportion rate type for premium payments. More details about Prorata Type Configuration in the description of the Policy Admin configurations section, in the Main Info tab.
Write off	Set the tolerance threshold for writing off payments. More details about Write off in the description of the Payments Schedule & Billing configurations section, in the Main Info tab.
Premium Invoice Generation	Configure the values to be used for automatic invoice generation. More details about Premium Invoice Generation in the description of the Payments Schedule & Billing configurations section, in the Main Info tab.
Tariff Type	Set the tariff type options. The option set values are: Per Coverage and Per Product .
Underwriting Type	Set the underwriting type options. The option set values are: Per Coverage and Per Product .

NOTE

Cloning a product is also a way to rapidly create a draft product, since every cloned product is registered in the system only in **Draft** status.

To create a product clone, please see the next section.

Cloning Insurance Products

NOTE

Every **cloned product** is registered in the system in **Draft** status, irrespective of the status of the original product. While the product is in **Draft** business status, the form is still editable and you can complete the product creation journey at a later time. For more details, see also the [Cloning Insurance Products](#) page.

Follow the below steps for cloning an insurance product:

1. In **Innovation Studio**, at the top left corner of your screen, click the main menu icon to open the main dropdown list.
2. From this main list, click **Insurance Product Factory**. A second panel opens, to the left.
3. Next, click **Insurance Products** to go to the **Insurance Products List** page.
4. From the list, select the **Insurance Product** that you want to clone and double-click it.
5. Once opened, go to the top right corner of the screen, and press the **Clone** button to launch the cloning.

Below, an example of the **Clone** button, next to Business Transactions button:



6. Next, the cloning pop-up opens. Use it to fill in a **Name** and an **Insurance Product Code** for the new product. After cloning, the product view opens and you can edit it - the product being in **Draft** status.

Modifying Insurance Products

NOTE

When your product is in **Active** status, changing it may be done only by adding a **new version**. If this is your case, follow the details from the next section, about versioning.

Follow the below steps for modifying an insurance product that is in **Draft** status:

1. In **Innovation Studio**, at the top left corner of your screen, click the main menu icon to open the main dropdown list.
2. From this main list, click **Insurance Product Factory**. A second panel opens, to the left.
3. Next, click **Insurance Products** to go to the **Insurance Products List** page.
4. From the list, select the **Insurance Product** that you want to edit and double-click it. Once opened, use the form to edit your product.

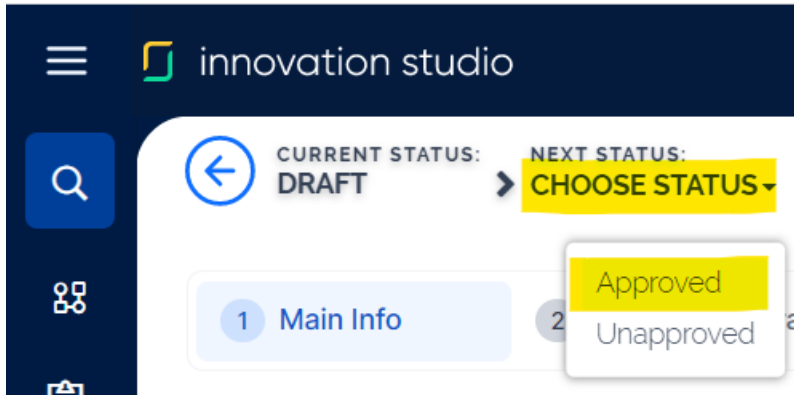
Versioning Insurance Products

Follow the below steps for versioning an insurance product:

1. In **Innovation Studio**, at the top left corner of your screen, click the main menu icon to open the main dropdown list.
2. From this main list, click **Insurance Product Factory**. A second panel opens, to the left.
3. Next, click **Insurance Products** to go to the **Insurance Products List** page.
4. From the list, select the product that you want to version and double-click it.
5. Once opened, go to the top right corner of the screen, and press the **Plus** button to launch the versioning process. The product becomes editable.
6. Use the form to make your adjustments.

7. Once finished, use the **status picker** at the top left corner of the screen to change the status of the newly created version from **Version Draft** to **Approved**.

Below, an example of the status picker:



8. After versioning, the product view opens and you can see your adjustments. You can also check the versioning log in the product's **History** tab.

NOTE

For deploying changes on existing products, you can also choose to import the changes in a **Data Config Definition** file, as this functionality has the versioning mechanism embedded. For more details, see also the Importing Insurance Products page.

Deleting Insurance Products

IMPORTANT!

Pay attention to the fact that some policies might include the product that you want to delete.

Follow the below steps to delete an **Insurance Product** record:

1. In **Innovation Studio**, at the top left corner of your screen, click the main menu icon to open the main dropdown list.
2. From this main list, click **Insurance Product Factory**. A second panel opens, to the left.
3. Next, click **Insurance Products** to go to the **Insurance Products List** page.
4. From the list, select the product record and click **Delete**, at the top right corner of the page.

Importing Product Data

This functionality lets you add product data on a destination environment by using the **FintechOSData Import Templates**. The most important use case for this functionality is a fast bulk import into the system of numerous new insurance products - with all imported products being registered in **Draft** status. But this is not the only use case. For more details, consult the [Importing Insurance Products](#) page.

Here are the details for importing **Product Data** by using the available **Data Import Templates** configured for the **Insurance Product Factory** solution:

1. In **Innovation Studio**, at the top left corner of your screen, click the main menu icon to open the main dropdown list.
2. From this main list, click **Evolutive Data Core**. A second panel opens, to the left.
3. Click **Data Import Templates** to open the **Data Import Templates List** page.
4. From the list, select the **Data Import Template** that you want to update and double-click it. Once opened, scroll down to the **List Of Data Imports** section, and press **Insert** to upload your file. After import, you can see the new import added in the list and you also check the versioning log in the product's **History** tab.

IMPORTANT!

Templates are different depending on what kind of data needs to be added. Check the [Importing Insurance Products](#) page, for more details about the templates for every import category.

Interrogating Product Data

The solution lets you interrogate product data by using API calls. For a specified insurance product, or product item (coverage), you can:

- get the premium calculation amount (price) valid at a certain date.
- get the underwriting decision result valid at a certain date.
- get all its attached tariff and underwriting formulas.

For more details, start from the [Insurance Product Factory Endpoints](#) page.

Access Product Interrogation History

This functionality lets you access the product interrogation history on a destination environment. In **Innovation Studio**, in the **Product Interrogation History** section, you have an overview of all the API calls registered in your system.

To view your **Product Interrogation** records, follow the steps below:

1. In **Innovation Studio**, at the top left corner of your screen, click the main menu icon to open the main dropdown list.
2. From this main list, click **Insurance Product Factory**. A second panel opens, to the left.
3. Next, click **Product Interrogation History** to go to the **Product Interrogation History** page.

On the **Product Interrogation History** page:

- To inspect a record from the grid, double-click it.
- To export a record from the grid, select it and press **Export**, at the top right corner of your screen.

For more details, consult the [Product Interrogation History](#) page.

Rules Impacting Products

The following is the behavior of the **Insurance Product Factory** solution, depending on the product business statuses:

- When the **Product** is in the **Approved** status - items or features **cannot be inserted/ edited/ modified/ deleted** from the product without using the **versioning functionality**.
- When the **Product** is in one of the following statuses: **Version Unapproved**, **Unapproved**, **Version Closed**, and **Closed** - items or features **cannot be inserted/ edited/ modified/ deleted** from the product.
- When the product is in one of the following statuses: **Draft** or **Version Draft** - items or features **can be inserted/ edited/ modified/ deleted** from the product.

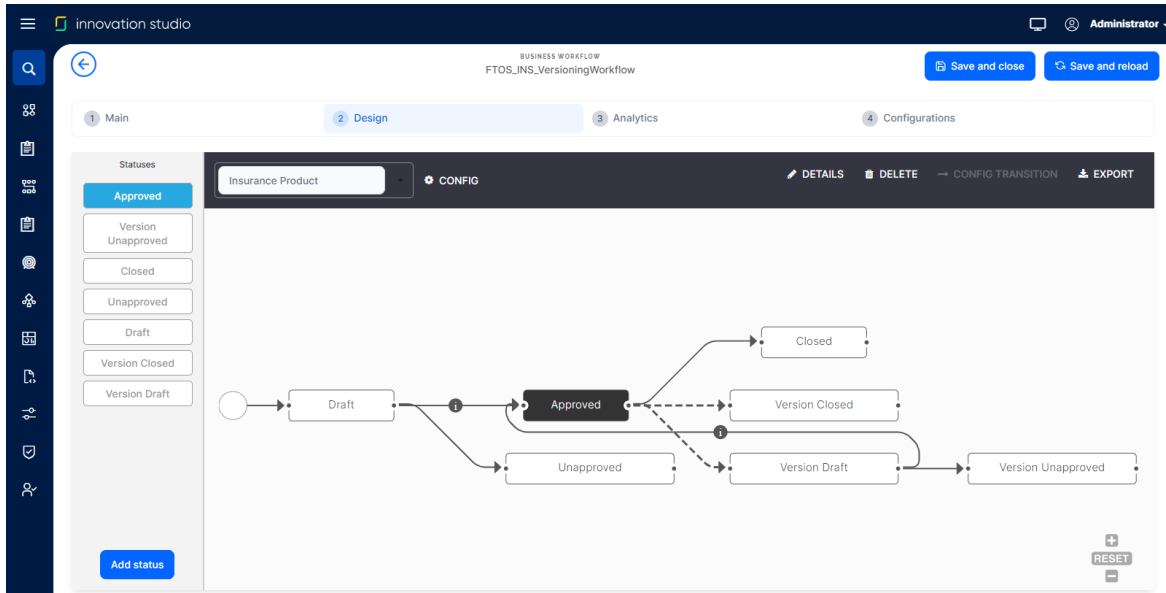
Product Life Cycle

The **Insurance Product Factory** solution handles different types of changes affecting an insurance product during its lifetime. This is accomplished by transitioning the product through different business states (from **Draft** to **Closed**) and also by versioning it. The solution also logs these product life cycle details in the product history tab, for further analytical use.

Product versioning helps you to stay relevant for your customers. It helps you to differentiate the product sold at any point in time. Additionally, it facilitates performance analysis: you can easily compare premiums and claims, and determine if that particular product version is profitable.

Product versioning is performed by using the **FintechOS FTOS_INS_VersioningWorkflow**. This is a master business workflow, used to add the versioning functionality on any desired entity - including the entities used by the insurance solutions.

Below, an example of the **FTOS_INS_VersioningWorkflow** business workflow used by the **Insurance Product Factory** solution, as it is displayed on **Innovation Studio**:



Product Behavior

The following are the behaviors - characteristic to any product, managed by this solution:

- Every product (or product version) starts in **Draft** status, goes through a refinement process - that concludes with an approval step, before going live.
- Once a product is live, its settings **can no longer be modified**.
- If you want to update an **Active** product, you must create a new product version. After editing, you must manually approve the new product version; use the status picker to pick the **Approved** status. For more details, see the **Versioning Insurance Products** section, on the [Managing Insurance Products](#) page.
- When you create a new product version, the current version is retired.

- Products in **Closed** status cannot be edited (versioned) - by you or the system.
- Only one version of a product can be live at one time.
- The system also can automatically adjust a product, based on product data received through a **Data Import Template**. All adjustments are automatically integrated into a new **Version Draft** of the product (that must be manually approved), and logged into the product history. For more details, see the [Importing Insurance Products](#) page.
- Only one **Draft Version** can be active at one time. Close any current draft version, before opening another one.

Product States and State Transitions

States Descriptions

Status name	Description
Draft	Initial state for any insurance product registered in the system. The draft products can be edited.
Unapproved	When a draft product is canceled.
Approved	Ongoing state - for live products. Products can't be edited.
Closed	Final state. The product is retired. The product cannot be moved from this state to any other states.
Version Draft	When the selected product becomes editable.
Version Unapproved	When the updates on the product are not approved.
Version Closed	When the product's draft version is closed.

Product State Transitions

Transition	Description
_Draft	Initial state - for any insurance product issued into the system.
Draft_Approved	When a product goes from draft to live status. Live products can't be edited.
Draft_Unapproved	When the proposed product is canceled.
Approved_Closed	When a live product is no longer active.

Transition	Description
Approved_Version Draft	When a new version is opened for a live product. The version is editable.
Approved_Version Closed	When the opened version is closed.
Version Draft_ Approved	When the opened version is approved and becomes the new version for the selected product.
Version Draft_ Version Unapproved	When the opened version is not approved. The product goes live without being modified.

Business Workflow

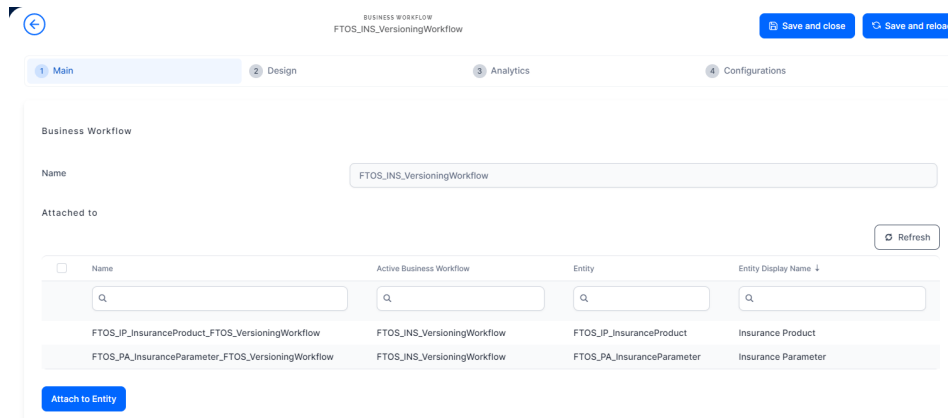
To access the **Insurance Product Factory** versioning business workflow in **Innovation Studio**, take the following steps:

1. In **Innovation Studio**, navigate down the main menu to **Automation Blocks**.
2. Click **Automation Blocks** and next, click Business Workflows, to open the **Business Workflows Menu**, on the left.
3. Click **Business Workflows Designer** to open the **Business Workflows List** page.
4. From the list, select and double-click the **FTOS_INS_VersioningWorkflow** record to inspect the business workflow attached to the [Insurance Product](#) entity.

When the form opens, you can see the main information about the workflow, its statuses and transitions.

Below, an example of the business workflow details, as they are

displayed in **Innovation Studio**:



HINT For details about the versioning steps, go to the [Managing Insurance Products](#) page > **Product Operations** chapter > **Versioning Insurance Products** section.

Insurance Perils And Conditions

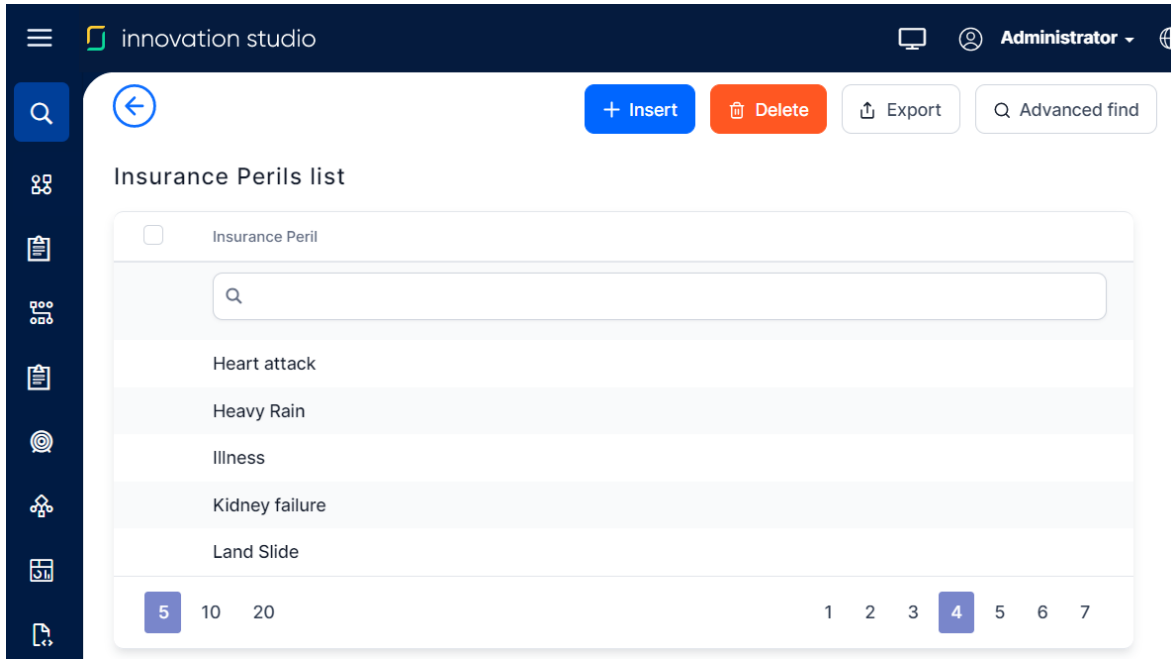
An **Insurance Peril**, or condition - for example **Earthquake**, **Car accident**, **Tornadoes**, **Theft**, **Death**, or **Disability**, informs about the type of coverage for a particular insurance product, or product item (coverage). The **Insurance Product Factory** allows you to define **Perils** independently from **Products**, so that they can be used in conjunction with multiple [insurance products](#). Depending on your insurance product, you can attach one or multiple perils, or conditions to it.

Insurance Perils View

In **Innovation Studio**, in the **Insurance Perils/ Conditions** section, you have an overview of all the insurance perils registered in your system - your nomenclature of perils, or conditions. This is an all-inclusive view; yet, you can also use the **Search**

option in order to find a specific peril record.

Below, you can see an example of an **Insurance Perils List**:



Follow these steps to view your **Insurance Perils**:

1. In **Innovation Studio**, at the top left corner of your screen, click the main menu icon to open the main dropdown list.
2. From this main list, click **Insurance Product Factory**. A second panel opens, to the left.
3. Next, click **Insurance Perils** to go to the **Insurance Perils List**.

On the **Insurance Perils List** page:

- To inspect a record from the grid, double-click it.
- To add a new record, click **Insert**, at the top right corner of the page.
- To edit a record from the grid, double-click it and make your updates. Next, click **Save and close**.
- To delete a record from the grid, select it and click **Delete**, at the top right corner of the page.

HINT

You can export one or more records by pressing **Export**, at the top right corner of your screen.

Insurance Peril Insert Form

The following fields are available for configuring an **Insurance Peril**:

Field	Description
Name	Insert the name of the insurance peril.
Max Notify Period	Add the maximum period for notification of peril.
Event Count Limit	Add the number of events covered by the policy.
Implicit Reserve	Add the amount of the prudential reserve to be deposited for the current policy.
Implicit Reserve Currency	Add the currency of the prudential reserve.

Below, an example of the **Insurance Peril** insert form:

The screenshot shows the 'Insurance Peril' insert form. At the top, there is a navigation bar with a back arrow on the left and three buttons: 'Save and close', 'Save and reload', and 'Save and new'. Below the navigation bar is a header 'Insurance Peril'. The main form area contains five input fields: 'Name', 'Max Notify Period', 'Event Count Limit', 'Implicit Reserve', and 'Implicit Reserve Currency'. The 'Implicit Reserve Currency' field has a dropdown arrow and an edit icon.

Insurance Perils Operations

The **Insurance Perils** functionality allows you to:

Create Perils

In order to create an **Insurance Peril**, follow the steps below:

1. On the **Insurance Perils List** page, click **Insert**, at the top right corner of the page. The **Insurance Peril** form opens.
2. Use the form to fill in your details (see the form description, in the section above).
3. Click **Save and close**.
4. Repeat these steps for each peril that you need to configure.

Modify Perils

In order to modify an **Insurance Peril**, please follow the next steps:

1. Open the **Insurance Perils List** page.
2. From the list, choose the desired **Insurance Peril** record and double-click to open the record.
3. Use the form to make your adjustments.
4. Click **Save and close**, at the top right corner of your screen.

Delete Perils

To delete an **Insurance Peril** from the grid, please follow the next steps:

1. Open the **Insurance Perils List** page.
2. From the list, select the **Insurance Peril** record that you want to delete and then click **Delete**, at the top right corner of the page.

Insurance Product Types

An **Insurance Product Type** - for example **Health Insurance**, **Property Insurance**, **Travel Insurance**, or **Pet Insurance**, helps you classify the insurance products you create. The **Insurance Product Factory** allows you to define **Product Types** independently, so that they can be used in conjunction with multiple **Insurance Products**. From the business perspective, **Product Types** help you sort out product records faster, and also make it easier to gather data for reporting & analysis.

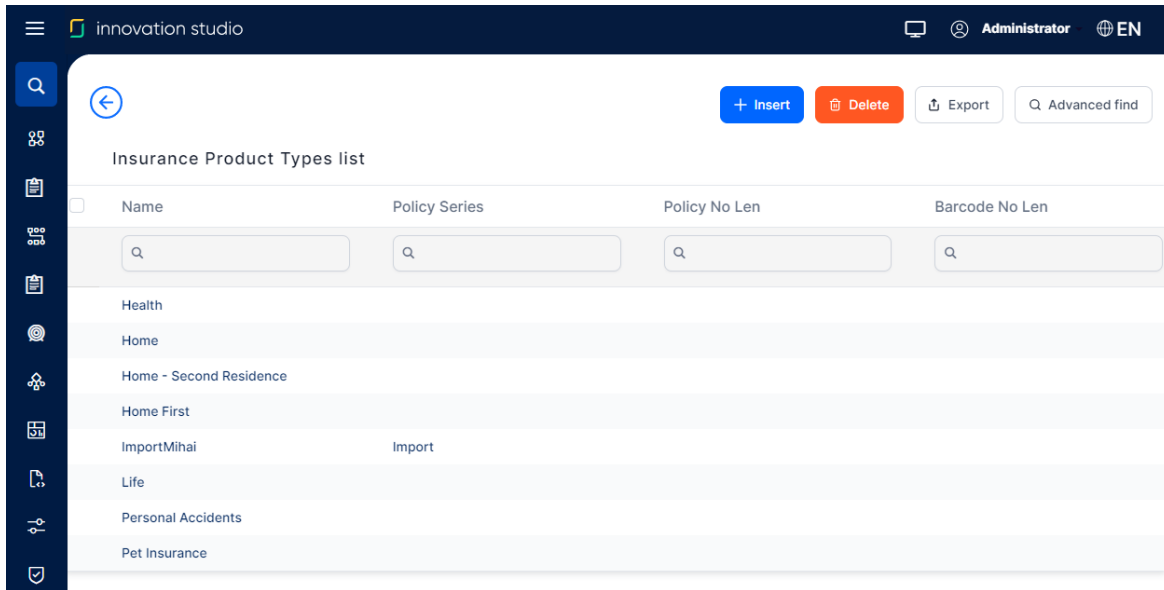
NOTE

In order to create a **Product**, you must indicate a **Product Type**! Either select one from the list of existing product types or insert a new one, when configuring the product.

Insurance Product Types View

In **Innovation Studio**, in the **Insurance Product Types** section, you have an overview of all the types registered in your system - your nomenclature of Insurance Product Types. This is an all-inclusive view; yet, you can also use the **Search by** option (in the list header) in order to find a certain record.

Below, you can see an example of an **Insurance Product Types List**:



Follow these steps to view your **Insurance Product Types**:

1. In **Innovation Studio**, at the top left corner of your screen, click the main menu icon to open the main dropdown list.
2. From this main list, click **Insurance Product Factory**. A second panel opens, to the left.
3. Next, click **Insurance Product Types** to go to the **Insurance Product Types List**.

On the **Insurance Product Types List** page:

- To inspect a record from the grid, double-click it.
- To add a new record, click **Insert**, at the top right corner of the page.
- To edit a record from the grid, double-click it and make your updates. Next, click **Save and close**.
- To delete a record from the grid, select it and click **Delete**, at the top right corner of the page.

HINT

You can export one or more records by pressing **Export**, at the top right corner of your screen.

Insurance Product Type Insert Form

The following fields are available for configuring an **Insurance Product Type**:

Insurance Product Type Section

This is the header section and it contains the following fields:

Field	Description
Name	Insert the name of the insurance product type.
Policy series	Leave blank - the series of the insurance policy is presently configured through a sequencer.
Policy no len	Leave blank - the number of digits of the insurance policy number is presently configured through a sequencer.
Barcode no len	Leave blank.

Below, an example of the form's header section:

The screenshot shows the 'Insurance Product Type' form in the Innovation Studio interface. The top navigation bar includes the 'innovation studio' logo, a user profile icon labeled 'Administrator', and a language selector 'EN'. Below the navigation bar, there are three action buttons: 'Save and close', 'Save and reload', and 'Save and new'. The main content area is titled 'Insurance Product Type' and contains a form with the following fields:

- INSURANCE PRODUCT TYPE** (Section Header)
- Name**: A text input field.
- Policy Series**: A text input field.
- Policy No Len**: A text input field.
- Barcode No Len**: A text input field.

Insurance Products Section

This is the middle section - inside this grid you can see all the products associated with that particular insurance product type.

Below, an example of the form's middle section - the example **Type** being included in two insurance products:

INSURANCE PRODUCTS

<input type="checkbox"/>	Name	Insurance Product Code	Start Date	End Date	Insurance Product Type	Label
<input type="checkbox"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
	Household Star Insurance HSI	HSI	20/10/2021	31/12/2023	Home First	Draft
<input checked="" type="checkbox"/>	Household Property Insurance	TCOH	20/10/2021	31/12/2023	Home First	Approved

NOTE
 By clicking **Insert** on the grid, you can create a new product, pre-filled with details from that specific **Product Type**.

Policy Alteration Type Section

This is the bottom section - inside this grid you can see all the **Alteration Types** associated with that particular product type. The available policy alteration options are: **Change frequency**, **Change payment type**, or **Update coverage**.

Below, an example of the form's bottom section:

POLICY ALTERATION TYPE

Name

Name

Change frequency

Change payment type

Update coverage

For more details, scroll down to the [Attaching Policy Alteration Types](#) section.

Insurance Product Types Operations

The **Insurance Product Type** functionality allows for the following operations:

Creating Types

In order to create an **Insurance Product Type**, follow the steps below:

1. On the **Insurance Product Types List** page, click **Insert**, at the top right corner of the page. The **Insurance Product Type** form opens.
2. Provide a **Name** for the type that you create.
3. Click **Save and close**.
4. Repeat the steps above for each product type that you need to configure.

Modifying Types

In order to modify an **Insurance Product Type**, please follow the next steps:

1. Open the **Insurance Product Types List** page.
2. From the list, choose the desired **Insurance Product Type** record and double-click to open the record. Use the form to make your adjustments.
3. Click **Save and close**, at the top right corner of your screen.

IMPORTANT!

Before modifying a certain product type, take into account the products already attached to that type.

Deleting Types

To delete an **Insurance Product Type** from the grid, please follow the next steps:

1. Open the **Insurance Product Types List** page.
2. From the list, select the **Insurance Product Type** record that you want to delete and then click **Delete**, at the top right corner of the page.

IMPORTANT!

Before deleting a certain product type, take into account the products already attached to that type.

Attaching Policy Alteration Types

There are times when you need to alter a policy - for example, if the policyholder decides to pay through an online payment processor instead of direct debit, an update of the payment type must be made on the policy. In **Insurance Product Factory**, you use the **Policy Alteration** option to indicate that these kind of alterations (updates) are allowed on the policies belonging to a particular insurance type.

In order to attach an **Alteration Type** to an **Insurance Product Type**, please follow the next steps:

1. On the **Insurance Product Types List** page, choose the desired **Product Type** and double-click it to open the **Insurance Product Type** form.
2. Scroll down to the **Alteration Type** grid.

Below, an example of the grid, with no **Policy Alteration Type** attached:

POLICY ALTERATION TYPE

Name

No data

3. Press **Insert existing**. Next, choose an alteration from the list and click **Ok**. You can select from the available options: **Change frequency**, **Change payment type**, or **Update coverage**.

Below, an example of the grid, with alteration types inserted:

The screenshot shows a user interface for selecting alteration types. At the top right, there are four buttons: '← Cancel', '× Remove', '+ Insert', and '✓ Ok'. Below these buttons is a search bar with a magnifying glass icon. Underneath the search bar, there is a list of alteration types: 'Change frequency', 'Change payment type', and 'Update coverage'. Each item in the list has a small square checkbox to its left, which is currently unchecked.

4. Click **Save and close**.

Insured Object Types

An **Object** groups together individual properties (dimensions) that transcend an insurance product. For example, for a **Motor** object, you might want to define **Brand**, **Model**, and **Year** dimensions, amongst others. On the other hand, for a **Person** (as a particular Life insurance object type), you might want to define **Age**, **Weight**, **Medical Conditions**, and **Employment** dimensions.

The functionality allows you to create diverse object types, from types representing very concrete insurable objects - such as house objects, or car objects, to health and life objects, and even more abstract objects - such as cyber insurance objects. Additionally, your objects are independent from [products](#), so that you can easily operate with and maintain them. (See also the [Insured Object Types Operations](#) section, at the bottom of the page.)

Once defined, and **Approved**, you can embed the same object into different insurance products, and offer it to different audiences. For example, an apartment object can be included in two or more property insurance products, each of the products having different types of coverages, premium calculation formulas, and underwriting rules.

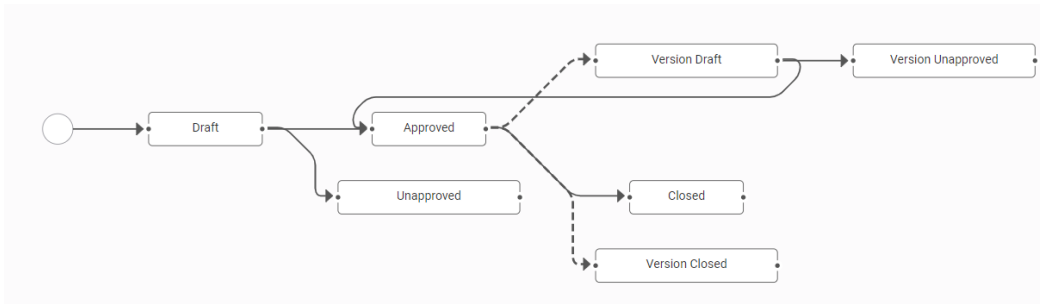
Preventing deletion of **Approved** objects is one important aspect of this feature and provides benefits, as follows:

- ensures that product **integrity** is maintained while the product is live.
- products must remain "as purchased" to **comply with regulations** - meaning you can't change a product once purchased by a customer, unless you advise them of the change, in advance, in writing.
- data is available for accurate **reporting** - underwriters need to compare like with like for their pricing models.

Insured Object Types Life Cycle

The **Insurance Product Factory** solution handles different types of changes affecting an insured object type during its lifetime. This is accomplished by transitioning the object through different business states (from **Draft** to **Closed**) and also by versioning it. The solution also logs these life cycle details in the object history tab, for further analytical use.

Below, an example of the **Insured Object Type** business workflow, including the versioning:



The available **Insured Object Type** business statuses are:

- Draft
- Approved
- Unapproved
- Closed
- Version Draft

- Version Unapproved
- Version Closed

IMPORTANT!
An insurance product can only be **Approved** if it has an attached **object type**!

Insured Object Types View

In **Innovation Studio**, in the **Insured Object Types** section, you have an overview of all the object records registered in your system - your collection of **Insured Object Types**. This is an all-inclusive view; yet, you can also use the **Search by** option (in the list header) in order to find a certain record.

Below, you can see an example of an **Insured Object Types List**, with objects filtered by their business status:

⏪

+ Insert
Delete
↑ Export
Q Advanced find

Insured Object Types list

<input type="checkbox"/>	Name	Object Type	Entity Mapping	Business Status
	<input type="text" value="Q"/>	<input type="text" value="Q"/>	<input type="text" value="Q"/>	<input type="text" value="Q Approved"/>
	2 Storey House	Property	FTOS_INSQB_InsuredOb...	Approved
	3 Storey House	Property	FTOS_INSQB_InsuredOb...	Approved
	Cat	Pet	FTOS_INSQB_InsuredOb...	Approved
	Cat - Gold Coverage	Pet	FTOS_INSQB_InsuredOb...	Approved
	Holiday House	Property	FTOS_INSQB_QuoteProp...	Approved
	Motor Luxury	Motor	FTOS_INSQB_InsuredOb...	Approved
	Penthouse	Property	FTOS_INSQB_InsuredOb...	Approved

Follow these steps to view your **Insured Object Types**:

1. In **Innovation Studio**, at the top left corner of your screen, click the main menu icon to expand the main dropdown list.
2. From this main list, click **Insurance Product Factory**. A second panel opens, to the left.
3. Next, click **Insured Object Types** to go to the **Insured Object Types List**.

On the **Insured Object Types List** page:

- To inspect a record from the grid, double-click it.
- To add a new record, click **Insert**, at the top right corner of the page.
- To edit a **Draft** record from the grid, double-click it and make your updates. Next, click **Save and close**.
- To delete a record from the grid, select it and click **Delete**, at the top right corner of the page.

HINT

You can export one or more records by pressing **Export**, at the top right corner of your screen.

Insured Object Type Insert Form

The following fields are available for configuring an **Insured Object Type**:

Insured Object Type Section

This is the header section and it contains the following fields:

Field	Description
Name	Insert the name of the object.
Object Type	Use the dropdown to select a type for the object. The available options are: motor , property , travel , personal accidents , and insuredObjectPet . (You can add more items to the option set, as needed.)
Insured Object Type Version	The version number is logged here, automatically.

Field	Description
Description	Describe your object.
Entity Mapping	Use the dropdown to indicate an entity. This mapping makes all attributes of that entity available for you, for creating object dimensions.

Below, an example of the form's header section, with some details filled in, for a property object:

The screenshot shows the 'Innovation Studio' interface. At the top, it displays 'innovation studio' and 'Administrator' with a language selector set to 'EN'. Below this, there are status indicators: 'CURRENT STATUS: Draft' and 'NEXT STATUS: Choose status'. A 'Save and reload' button is visible. The main content area is titled 'Insured Object Type' and contains several input fields:

- Name:** 1 Storey House
- Object Type:** Property (dropdown menu)
- Insured Object Type Version:** 1
- Description:** Property insurance for one-storey houses.
- Entity Mapping:** FTOS_INSQB_InsuredObjectProperty (dropdown menu)

Dimensions Section

This is the bottom section - where you can see all the **Dimensions** associated with your object. Inside this grid, the available operations are **Insert**, **Delete**, **Export**, and **Refresh**. After adding the dimensions, you can order them by using the **order index** column. You can drag and drop rows to reorder them - this also sets the order index for how dimensions are displayed further, on the policy form.

Below, an example of the form's bottom section, with some dimensions added, for a property object:

Dimensions

+ Insert X Delete Export Refresh

<input type="checkbox"/>	Name	Display Name	Entity Attribute	Details	Is Mandatory	Order Index ⌵
<input type="text" value="Q"/>	<input type="text" value="Q"/>	<input type="text" value="Q"/>	<input type="text" value="Q"/>	<input type="text" value="Q"/>	(All) ▼	<input type="text" value="Q"/>
	addressId	addressId	addressId	addressId	-	9
	alarmSystemCon...	alarmSystemCon...	alarmSystemCon...	alarmSystemCon...	-	22
	annexSurface	annexSurface	annexSurface	annexSurface	-	39
	ApartmentNo	ApartmentNo	ApartmentNo	ApartmentNo	-	52
	BuildingNo	BuildingNo	BuildingNo	BuildingNo	-	73
	constructionTypeld	constructionTypeld	constructionTypeld	constructionTypeld	-	90
	constructionYear	constructionYear	constructionYear	constructionYear	-	103
	creditType	creditType	creditType	creditType	-	120

NOTE
 By clicking **Insert** on the grid, you can add **attributes**, as dimensions, **only from the entity** you have previously mapped to your object.

Follow the below steps, to add an object dimension:

1. In the **Dimensions** grid, click **Insert**. The dimension insert form opens.
2. Use the form to fill in the available fields:

Field	Description
Name	Indicate a name for the dimension. For example: brand, model, constructionYear, buildingType etc. This is a mandatory field.
Display name	Indicate a display name for the dimension. For example: Brand, Model, Construction year, Building type etc. This is a mandatory field.

Field	Description
Entity attribute	Use the arrow/ picker to select the attribute correlated with the dimension. You can only select attributes from the entity that you mapped when previously configuring the object. This is a mandatory field.
Details	Area for details regarding the added dimension. Not a mandatory field
Is Mandatory	Set to true if the current dimension is mandatory.

3. Click **Save and close** or **Save and new**, if you want to add another dimension to your object.

Below, an example of the dimension insert form, with some details filled in, for a property object:

HINT

You do not need to add the **addressId** dimension since this is added by default, for every object.

Insured Object Types Operations

The **Insured Object Types** functionality allows for the following operations:

Creating Objects

In order to create an **Insured Object Type**, follow the steps below:

1. On the **Insured Object Types List** page, click **Insert**, at the top right corner of the page. The **Insured Object Type** form opens.
2. Use the form to create your **Object**. (See also the details above.)
3. Fill in the fields.
4. Select the entity mapping.
5. Select entity attributes to indicate object dimensions.
6. Click **Save and close**.
7. Repeat the steps above for each object that you need to configure.

Modifying Objects

NOTE

When your object is in **Active** status (namely, it is attached to a **Product**), changing it may be done only by adding a **new version**. If this is your case, follow the details from the next section, about versioning.

Follow the below steps for modifying an object that is in **Draft** status:

1. Open the **Insured Object Types List** page.
2. From the list, choose the desired **Insured Object Type** record and double-click to open the record.

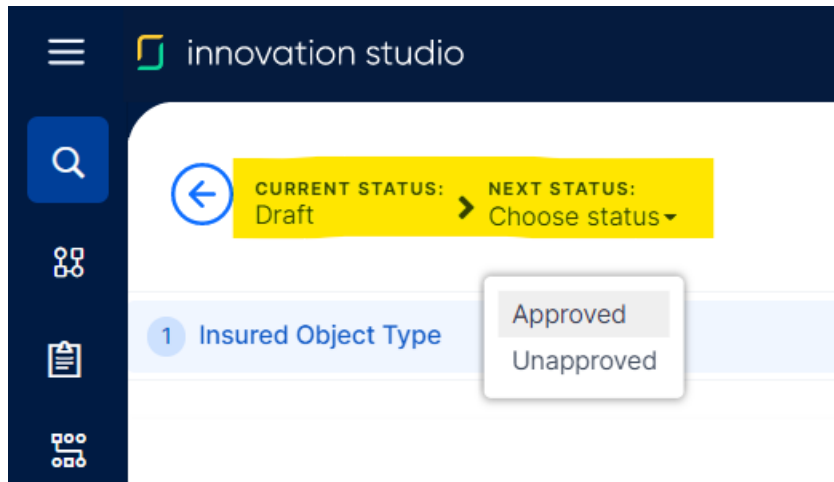
3. Use the form to make adjustments to your object.
4. Click **Save and close**, at the top right corner of your screen.

Versioning Objects

Follow the below steps for versioning an **Insured Object Type** in **Approved** status (namely attached to a **Product**):

1. Open the **Insured Object Types List** page.
2. From the list, choose the desired **Insured Object Type** record and double-click to open the record.
3. Once opened, go to the top right corner of the screen, and press the **Plus** button to launch the versioning process. The object becomes editable.
4. Use the form to make your adjustments.
5. Once finished, use the **status picker** at the top left corner of the screen to change the status of the newly created version from **Version Draft** to **Approved**.

Below, an example of the status picker:



6. After versioning, the object view opens and you can see your adjustments.
You can also check the versioning log in the object's **History** tab.
7. Click **Save and close**, at the top right corner of your screen.

Deleting Objects

IMPORTANT!

You can only delete **Object Types** in **Draft** or **Unapproved** status.

To do so, please follow the next steps:

1. Open the **Insured Object Types List** page.
2. From the list, select the **Insured Object Type** record and click **Delete**, at the top right corner of the page.

NOTE

You cannot delete object types in status **Closed** and **Approved Object Types**, assigned to approved products.

Product Interrogation History

The **Insurance Product Factory** allows you to consult the API interrogations history for your insurance products, in **Innovation Studio**. The solution stores all the API requests and their corresponding responses for the following APIs:

- [FTOS_IP_GetUWRulesResultAPI](#),
- [FTOS_IP_PremiumAmountAPI](#),
- [FTOS_IP_ProposalConfigPremiumCalculation](#).

The following types of product data are available in the **Product Interrogation History**:

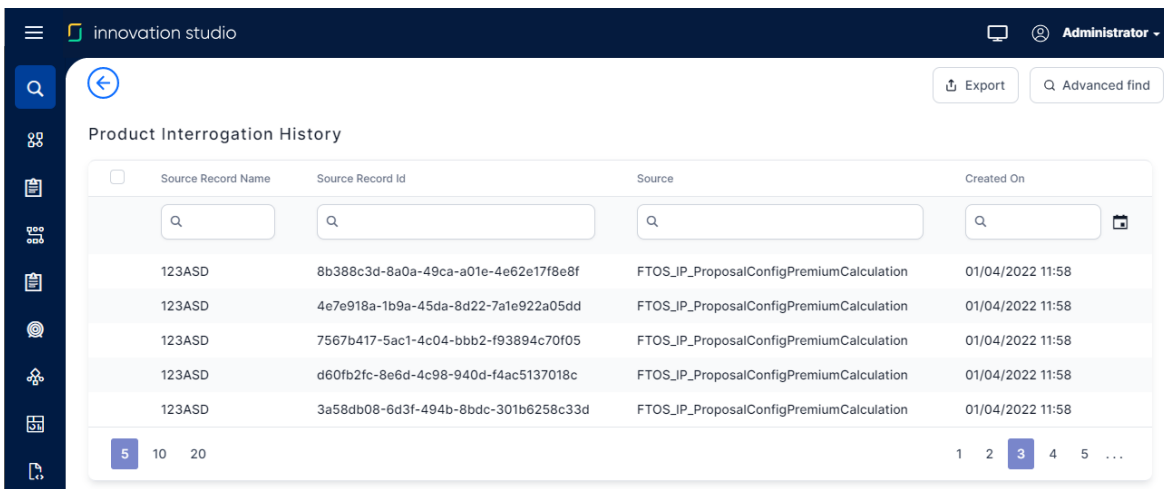
- Prices requests including from multiple proposals, or multiple cards, and their results broken down on each formula step.
- Underwriting decision results, broken down on each formula step.

NOTE
The **Product Interrogation History** section is automatically updated, and you cannot add or delete records.

Product Interrogation History View

In **Innovation Studio**, in the **Product Interrogation History** section, you have an overview of all the API interrogations registered in your system. This is an all-inclusive view; yet, you can also use the **Search** option in order to find a specific product interrogation record.

Below, you can see an example of an **Product Interrogation History List**:



To view your **Product Interrogation** records, follow the steps below:

1. In **Innovation Studio**, at the top left corner of your screen, click the main menu icon to open the main dropdown list.

2. From this main list, click **Insurance Product Factory**. A second panel opens, to the left.
3. Next, click **Product Interrogation History** to go to the **Product Interrogation History** page.

On the **Product Interrogation History** page:

- To inspect a record from the grid, double-click it.
- To export a record from the grid, select it and press **Export**, at the top right corner of your screen.

IMPORTANT!

The history of product interrogations is only available if the system parameter **FTOS_IP_ProductInterrogationHistory_Enablelog** value is set to **1**.

Solution Configurations

The **Insurance Product Factory** solution enables you to create, adjust and maintain **Insurance Products**. Check the following pages to find out more about how this solution works:

- [Insurance Product Factory Endpoints and API History](#)
- [Insurance Product General Operations](#)
- [Cloning Insurance Products](#)
- [Importing Insurance Products](#)
- [Configuring Payment Types](#)
- [Configuring Insured Object Types](#)
- [Configuring Lines Of Business](#)
- [Insurance Formulas](#)
- [Insurance Product Factory Digital Assets](#)

Configuration FAQs

- Insurance product configuration starts by adding a new record into **FTOS_IP_InsuranceProduct** entity, with general details about the product: name, code, settings influencing other flows' behavior on **Policy Admin** or **Billing** solutions.
- The coverages and the modules are saved in the same entity - that is the **FTOS_IP_InsuranceProductItem** entity, a coverage being always a parent of a module. For that reason:

- A coverage is always linked to a product by **insuranceProductId** lookup.
- A module doesn't have the **insuranceProductId** lookup filled in, but instead it has the **itemParentId** lookup filled in with one of the existing "parent" coverages.
- The **FTOS_IP_InsuranceRisk** entity stores the general peril nomenclature table. But, since a peril can have a different configuration in the context of a certain product (like different limits), you must also use the **FTOS_IP_CoveredRisk** to create specific insurable perils, out of the general ones.
- A covered peril can be associated on multiple modules, and a module can have multiple covered perils. The **FTOS_IP_InsuranceCoveredRisk** entity is used to create the necessary links between them.
- In order to calculate the premium or get an underwriting score result, formulas can be associated to a product, or each coverage of a product. Pay attention to the fact that only one formula can be attached to a product, or coverage.
- For underwriting rules, the **context type** attribute is required, since underwriting rules can be divided in different categories, defined in **FTOS_IP_UnderwritingContextType**. Pay attention to the fact that only one formula for underwriting can be associated on a product, or coverage, on a specific context.

HINT

The **Insurance Product Factory** maintains the underlying data in a **consistent data model**. Consequently, you have a reliable data model for your insurance product portfolio that you can reference when you build your digital journeys. This allows you to manage your product portfolio at will, without having to re-code your digital journeys every time an insurance product is added, updated, or retired.

Insurance Product Factory Endpoints

There are cases when you need to access the product data or **Insurance Product Factory** functionalities in a way alternative to the one available as a user journey, in

Innovation Studio. The solution allows you to make API calls regarding different product-related aspects. For interacting with defined products, the following endpoints are available:

- [FTOS_IP_PremiumAmountAPI](#) - for getting prices on a specified insurance product, or product item (coverage).
- [FTOS_IP_GetUWRulesResultAPI](#) - for getting the underwriting decision for a specified insurance product, or product item (coverage).
- [FTOS_IP_GetProductFormulasStructuresAPI](#) - for getting all the tariff and underwriting formulas attached to a specified insurance product, or product item (coverage).

Apart from the above endpoints, the following are other endpoints available with the **Insurance Product Factory** solution:

FTOS_IP_CALC_TestPremiumAmount

Script: FTOS_IP_CALC_TestPremiumAmount

Description: This script is used during the Test Calculation step, to test the premium calculation formulas attached on insurance product coverages and return the result.

FTOS_IP_CALC_TestVerifyUnderwriting

Script: FTOS_IP_CALC_TestVerifyUnderwriting

Description: This script is used during the Test Calculation step, to test the underwriting rules on an insurance product and return the result.

FTOS_IP_CloneIP

Script: FTOS_IP_CloneIP

Description: This script gets all the configurations attached to a specified insurance product and uses them to create a new clone product.

FTOS_IP_FormulaDataMapping_CheckExisting

Script: FTOS_IP_FormulaDataMapping_CheckExisting

Description: This script checks if there is a data mapping of the same type already added on an insurance formula.

FTOS_IP GetInsuranceProductCalculationDetails

Script: FTOS_IP_GetInsuranceProductCalculationDetails

Description: This script gets the insurance product calculation details.

FTOS_IP_GetInsuranceProductType

Script: FTOS_IP_GetInsuranceProductType

Description: This script gets the insurance product type details.

FTOS_IP_GetLineOfBusinessVA

Script: FTOS_IP_GetLineOfBusinessVA

Description: This endpoint calls the FTOS_IP_GetLineOfBusinessVA on demand script to retrieve the class of business, or class of business and category of business. It is used on FTOS_IP_LineOfBusiness form and FTOS_IP_LineOfBusinessSubtype form.

FTOS_IP_InsuranceProduct_ CheckUnderwritingType

Script: FTOS_IP_InsuranceProduct_CheckUnderwritingType

Description: This script gets the insurance product underwriting type.

FTOS_IP_InsuranceProductCoverFormulaUnd_ CreateDataMapping

Script: FTOS_IP_InsuranceProductCoverFormulaUnd_CreateDataMapping

Description: This script creates the data mapping for the UW formulas, or rules attached on coverages.

FTOS_IP_InsuranceProductFormula_CreateDataMapping

Script: FTOS_IP_InsuranceProductFormula_CreateDataMapping

Description: This script creates the data mapping for the UW formulas, or rules attached on product.

FTOS_IP_InsuranceProductItemFormula_CreateDataMapping

Script: FTOS_IP_InsuranceProductItemFormula_CreateDataMapping

Description: This script creates the data mapping for the premium calculation formulas, or rules attached on coverages.

FTOS_IP_InsuranceProductItemFormulaUnd_CreateDataMapping

Script: FTOS_IP_InsuranceProductItemFormulaUnd_CreateDataMapping

Description: This script creates the data mapping for the underwriting calculation formulas, or rules attached on coverages.

FTOS_IP_ValidateIP

Script: FTOS_IP_ValidateIP

Description: This script is used in the cloning process, to check if the provided product name and code are not already used on another product.

FTOS_VersioningHelper_Edit_FetchEntity

Script: FTOS_VersioningHelper_Edit_FetchEntity

Description: This script handles server-side operations needed in the versioning process, when evaluating whether the versioning option should be available on a record or not.

HINT

The **Insurance Product Factory** endpoints are listed in **Innovation Studio**, in the

Endpoints List page. To find them, follow this path **Innovation Studio** main menu > **Advanced** > **Endpoints** > **Endpoints list**.

Get Formulas Structures API

Use this API to get the **formulas structure** for an insurance product. A structure is composed from all the input parameter **keys** expected by the all the formulas attached to the specified insurance product. Depending on that product configuration, the structure looks different from a product to another (see the example below).

Product 1's configuration is different from Product 2's configuration.

Product 1	Product 2
<ul style="list-style-type: none"> • tariff type - per product, • underwriting rules - per coverage, • underwriting context type set to medical underwriting. 	<ul style="list-style-type: none"> • tariff type - per coverage, • underwriting rules - per product, • underwriting context type set to building underwriting.

The input parameter keys expected by the formulas attached to Product 1 are going to be different from the input parameter keys expected by the formulas attached to Product 2.

The API response includes the following details:

- the structure for the premium calculation formula, attached to the product (if the case),
- the structure for all the premium calculation formulas for each of the product items (coverages),
- the structure for the underwriting formula (rules), attached to the product (if the case),
- the structure for all the underwriting formulas for each of the product items (coverages),
- the tariff type set on the product,
- the underwriting type and the underwriting context type set on the product.

If necessary, in the API call you can specify a **Validity Date** and get the formulas structure that was valid for your product, at that date.

NOTE

If no underwriting **context type** is defined, then the API returns an empty array for the **uwRulesStructures** array.

Example

A user makes a call for the formula structure for a certain insurance product - indicated by a specified product code. The user wants to retrieve the formula structure that was valid at a certain date - respectively, 2021-05-21.

```

1 | let p = { "productCode": "BA_EMB" "validityDate": "2021-
2 |   05-21"
3 | }
   | ebs.callActionByNameAsync("FTOS_IP_
   | GetProductFormulasStructuresAPI", p) .then(function(e) {
   |   console.log(e) })

```

Request data parameters

The following data parameters must be included in the request:

Parameter	Description
productCode	The code of the insurance product.
validityDate	(Optional) The reference date, prior to the current date, for getting an earlier version of the product formulas structure. When this key is not provided, the API returns the current version of the formula structure. Accepted formats are: yyyy-mm-dd, dd/mm/yyyy, dd-mm-yyyy, or dd.mm.yyyy.

Response

This is an example of a response:

```

1  { "isSuccess": true, "errorMessage": null, "errorCode":
    null, "result": [ { "productCode": "BA_EMB",
      "tariffType": "perCoverage", "uwType": "perCoverage",
      "premiumCalculationStructures": [ { "itemCode": "HHR0",
        // this key is not available for tariff type = per
        product "formulaVersion": 1.0, "formulaStructure": [ {
          "key": "risks", "value": null, "masterType":
          "Collection", "subType": "Object", "objProps": "
          {"Risk\":6,\"Module\":6,\"Price\":2}" }, { "key":
          "propertyInsuredAmount", "value": null, "masterType":
          "SimpleType", "subType": "Decimal", "objProps": null }, {
          "key": "propertyConstructionYear", "value": null,
          "masterType": "SimpleType", "subType": "WholeNumber",
          "objProps": null } ] }, { "itemCode": "HA", // this key
          is not available for tariff type = per product
          "formulaVersion": 1.0, "formulaStructure": [ { "key":
          "buildingInsuredAmount", "value": null, "masterType":
          "SimpleType", "subType": "Decimal", "objProps": null }, {
          "key": "constructionYear", "value": null, "masterType":
          "SimpleType", "subType": "WholeNumber", "objProps": null
          } ] } ], "uwRulesStructures": [ { "contextType":
          "Underwriting", "itemCode": "HA", // this key is not
          available for uw type = per product "formulaVersion":
          1.0, "formulaStructure": [ { "key": "insuredAmount",
          "value": null, "masterType": "SimpleType", "subType":
          "Decimal", "objProps": null }, { "key": "seismicZone",
          "value": null, "masterType": "SimpleType", "subType":
          "Text", "objProps": null } ] } ] } ]
2  }

```

Response description:

Key	Description
Error code	Error code.
Error message	Error message.
isSuccess	Marks whether the request was successful or not.
result	Array of objects containing details about the prices.
productCode	The code of the insurance product.
tariffType	The Tariff Type defined on the product level - either <code>perProduct</code> or <code>perCoverage</code> .
uwType	The Underwriting Type defined on the product level - either <code>perProduct</code> or <code>perCoverage</code> .

Key	Description
premiumCalculationStructures	An array containing the input parameters for the premium calculation formulas attached to the specified product, or product items (coverages).
uwRulesStructures	An array containing the input parameters for the underwriting formulas attached to the specified product, or product items (coverages).

Error Messages

The following are the error messages that can be encountered while calling the **GetFormulasStructuresAPI**:

Code	Text	Description
ERR.IP.50201	ERR.IP.50201 - Invalid validity date format! Please, use yyyy-mm-dd or dd/mm/yyyy or dd-mm-yyyy or dd.mm.yyyy!	Invalid date format for validityDate input parameter.
ERR.IP.50202	ERR.IP.50202 - Invalid validity date!	Invalid date for validityDate input parameter.
ERR.IP.50203	ERR.IP.50203 - Invalid key request!	Missing parameters in the request or wrong parameter name.
ERR.IP.50204	ERR.IP.50204 - No active product identified!	No active insurance product found.
ERR.IP.50205	ERR.IP.50205 - Error! The //code of product// insurance product was not approved at the requested date!	No active insurance product found at the date specified in validityDate input parameter.

Endpoints

The **FTOS_IP_GetProductFormulasStructuresAPI** endpoint validates the API call request, searches for the specified product and returns all the expected input parameter keys for all the formulas attached to that insurance product.

Server Side Script Library

FTOS_IP_InsuranceProductAPIs

From this library, use the **GetFormulaInputParameters** function (described below). This function wraps all the functions necessary to validate and return the formulas structure results.

validateRequest

This function validates the request fields.

Input parameters: **inputData** - The object containing the keys needed to call the endpoint.

Output parameters: An **object** containing the following **keys**, for describing the result of the validation:

- **isSuccess** - true/ false - The boolean that shows whether the validation is successful.
- **errorMessage** - A null value or an error message as described in the [error messages list](#).
- **errorCode** - A null value or an error code as described in the [error messages list](#).
- **result** = An empty array [] or an array with details about the input parameters formula structure, as described in the [response description](#) section.

getPremiumFormulaStructure

This function gets the input parameters for the premium calculation formulas attached to the specified product, or product items (coverages). The function also uses other helper functions, implemented in the same library, to get the product details, the list of items, and the formulas attached on items.

Input parameters: **inputData** - The object containing the keys needed to call the function.

Output parameters: An **array** containing objects with the following keys:

- `productCode` - The code of the insurance product.
- `tariffType` - The tariff type defined on the product (either `perProduct` or `perCoverage`).
- `uwType` - The underwriting type defined on the product (either `perProduct` or `perCoverage`).
- `premiumCalculationStructures` - An array containing the following objects:
 - `object` containing details about the product, or product items (coverages),
 - `object` containing the **item code** (when tariff type = `perCoverage`), the **formula version number** and an **array** with objects for each **input parameter**.

getUWFormulaStructure

This function gets the input parameters for the underwriting formulas attached to the specified product, or product item - coverage (identified by item code, provided in the request). The function also uses other helper functions, implemented in the same library, to get product details and the underwriting formulas structure.

Input parameters: `inputData` - The object containing the keys needed to call the function.

Output parameters: An `array` containing objects with the following keys:

- `productCode` - The code of the insurance product.
- `uwRulesStructures` - An array containing the following objects:
 - `object` containing details about the product, or product items (coverages),

- **object** containing the **underwriting context type**, the **item code** (when underwriting type = **perCoverage**), the **formula version number** and an **array** with objects for each **input parameter**.

getFormulaStructure

This function concatenates the results of the **getPremiumFormulaStructure** and **getUWFormulaStructure** functions into a single array. The function is called inside the endpoint only if the request passes the validation - namely if the **isSuccess** key from the response of the **validateRequest** function is **true**.

Input parameters: **inputData** - The object containing the keys needed to call the endpoint **FTOS_IP_GetProductFormulasStructuresAPI**.

Output parameters: An **array** containing objects with the following keys:

- **productCode** - An object from the **getPremiumFormulaStructure** output parameters.
- **tariffType** - The tariff type defined on the product (either **perProduct** or **perCoverage**).
- **uwType** - The underwriting type defined on the product (either **perProduct** or **perCoverage**).
- **premiumCalculationStructures** - An object from the **getPremiumFormulaStructure** output parameters.
- **uwRulesStructures** - An object from the **getUWFormulaStructure** output parameters.

Get Premium Amount API

Use this API to get premium calculation results, such as:

- the premium amount for an insurance product (total premium amount),
- the premium amount for different product items (coverages) included in an insurance product,
- the premium amounts for each coverage, based on the premium coverage split percentages defined at the product level,
- the price result that was valid at a certain date, for a specified insurance product, or product item (coverage).

The **FTOS_IP_PremiumAmountAPI** endpoint runs the premium calculation formulas assigned to each product, or product item (coverage), simulating the **Test Scenario** functionality - available for users in **Innovation Studio**.

NOTE

The results are in line with the **Tariff Type** set at the product level. An insurance product can have the tariff type set either to **perProduct** or **perCoverage**.

For a valid API request, include all the formula **Input Keys** expected by the **Business Formulas** engine (see below). When needed, use the **Validity Date** key to get the price result that was valid at that date, for the specified product, or product item (coverage).

HINT

All the requests through this API and their responses can be saved into the **FTOS_IP_ProductInterogationHistory** entity. This action is available if the system parameter **FTOS_IP_ProductInterogationHistory_Enablelog** value is set to **1**.

Example

A user makes a request for calculating the price (premium amount) for two product items (coverages) configured on a property insurance product. The user wants the premium amount formula which was valid at a certain date -

respectively, 2021-05-21.

```

1 | let p = { "insuranceTypeName": "Home", "productCode":
    | "BA_EMB", "validityDate": "2021-05-21",
    | "sourceRecordName": "P7857", "sourceRecordId": "0AFE8D0B-
    | F423-44F4-9638-DEE696BF0B0E",
    | "premiumCalculationDetails": [{ "itemCode": "HHR0", //
    | this key is not available for tariff type = per product
    | "calculationDetails": { "risks": [{ "Risk": "Earthquake",
    | "Module": "M1", "Price": 0 }, { "Risk": "Land Slide",
    | "Module": "M1", "Price": 0 }, { "Risk": "Floods",
    | "Module": "M1", "Price": 0 }, { "Risk": "Windstorm",
    | "Module": "M1", "Price": 0 }], "propertyInsuredAmount":
    | 100000, "propertyConstructionYear": 1980 } }, {
    | "itemCode": "HA", // this key is not available for tariff
    | type = per product "calculationDetails": {
    | "buildingInsuredAmount": 50000, "constructionYear": 1980
    | } }]}
2 | }; ebs.callActionByNameAsync("FTOS_IP_
    | PremiumAmountAPI",p)
3 | .then(function(e){ console.log(e)
4 | })

```

Request Data Parameters

The following data parameters must be included in the request:

Parameter	Description
calculationDetails	Object containing the keys needed to run the formula. The object structure is used for testing the formula attached to the specified product item (coverage). The structure differs from item to item, based on the formula configuration.
insuranceTypeName	The name of an Insurance Type configured in the system and stored on the FTOS_IP_InsuranceType entity.
itemCode	The code of the product item (coverage). This key is not available for products which have the Tariff Type set to perProduct .
premiumCalculationDetails	Array with details to identify and run the formulas.
productCode	The code of the Insurance Product .

Parameter	Description
sourceRecordId	The Id of the record that the system is calculating the premium for.
sourceRecordName	Text identifying the record that the system is calculating the premium for.
validityDate	The reference date, prior to the current date, for calling an earlier version of the formula. This key is not mandatory. When it is not provided, the API calls the current version of formula. Accepted formats are: yyyy-mm-dd, dd/mm/yyyy, dd-mm-yyyy, or dd.mm.yyyy.

Response

This example contains calculation details for two coverages:

```

1  { "isSuccess": true, "errorMessage": null, "errorCode":
    null, "result": [ { "totalPremiumAmount": 20.9,
      "tariffType": "perCoverage", "premiumCalculationResults":
      [ { "itemCode": "HHR0", "premiumAmount": 20.4 }, {
        "itemCode": "HA", "premiumAmount": 0.5 } ],
      "formulaDetails": [ { "itemCode": "HHR0", // this key is
        not available for tariff type = per product
        "formulaResult": { "risks": [ { "Risk": "Earthquake",
          "Module": "M1", "Price": 0.0 }, { "Risk": "Land Slide",
          "Module": "M1", "Price": 0.0 }, { "Risk": "Floods",
          "Module": "M1", "Price": 0.0 }, { "Risk": "Windstorm",
          "Module": "M1", "Price": 0.0 } ],
          "propertyInsuredAmount": 100000.0,
          "propertyConstructionYear": 1980.0,
          "risksCalculatedPrices": [ { "Risk": "Earthquake",
            "Module": "M1", "Price": 0.0054 }, { "Risk": "Land
            Slide", "Module": "M1", "Price": 0.001 }, { "Risk":
            "Floods", "Module": "M1", "Price": 0.001 }, { "Risk":
            "Windstorm", "Module": "M1", "Price": 0.013 } ],
            "baseQuotation": 0.0204, "premiumAmount": 20.4000 } }, {
            "itemCode": "HA", // this key is not available for tariff
            type = per product "formulaResult": {
              "buildingInsuredAmount": 50000.0, "constructionYear":
              1980.0, "coefHomeAssistancePrice": 0.001,
              "premiumAmount": 0.500 } } ] } ]
2  }

```

Response description:

Key	Description
Error code	Error code.
Error message	Error message.
isSuccess	Marks whether the request was successful or not.
result	Array of objects containing details about the prices.
formulaDetails	Array with details - either the result returned by the formula attached on the product level OR the results returned by the formulas attached to product items (coverages), and their item codes.
premiumCalculationResults	Array with objects, containing the item code and the premium amount.
tariffType	The Tariff Type defined on the product level. The available options are either <code>perProduct</code> or <code>perCoverage</code> .
totalPremiumAmount	The total premium amount for the product.

Error Messages

The following are the error messages that can be received while calling the **Get Premium Amount API**:

Code	Text	Description
ERR.IP.50101	ERR.IP.50101 - Invalid validity date format! Please, use yyyy-mm-dd or dd/mm/yyyy or dd-mm-yyyy or dd.mm.yyyy!	Invalid date format for validityDate input parameter.
ERR.IP.50102	ERR.IP.50102 - Invalid validity date!	Invalid date for validityDate input parameter.
ERR.IP.50103	ERR.IP.50103 - Invalid request!	Missing parameters in the request.
ERR.IP.50104	ERR.IP.50104 - No active product identified!	No active insurance product found.
ERR.IP.50105	ERR.IP.50105 - Error! The //code of product// insurance product was not approved at the requested date!	No active insurance product found, for the date specified in the validityDate input parameter.

Code	Text	Description
ERR.IP.50106	ERR.IP.50106 - Product doesn't have any product items (coverages) configured!	The insurance product identified has no insurance items (coverages).
ERR.IP.50107	ERR.IP.50107 - Invalid insurance item code.	Invalid product item (coverage) code.
ERR.IP.50108	ERR.IP.50108 - sourceRecordName is mandatory for premium calculation!	Source record name is mandatory.
ERR.IP.50109	ERR.IP.50109 - sourceRecordId must be uniqueidentifier type!	If transmitted, it has to be a GUID.

Endpoints

The **FTOS_IP_PremiumAmountAPI** endpoint runs the formulas assigned to each product, or product item (coverage), in line with their defined tariff type.

For **perProduct** tariff type, the endpoint runs the formula assigned to the product and returns the total premium amount and, also, the premium amounts for the product items (coverage), according to the premium coverage split percentage set on each coverage.

For **perCoverage** tariff type, the endpoint runs the formulas assigned to each product item (coverage), returns the result for each coverage and, also, returns the total premium amount for the specified product, summing up the premium amounts of all coverages.

Server Side Script Library

FTOS_IP_InsuranceProductAPIs

From this library, for getting prices for the products, or product items (coverages), use the following functions:

validateRequest

This function validates the request fields.

Input parameters: `inputData` - The object containing the keys needed to call the endpoint.

Output parameters: An `object` containing the following **keys** to describe the result of the validation:

- `isSuccess` - true/ false - The boolean that shows whether the validation is successful.
- `errorMessage` - A null value or an error message as described in the [error messages list](#).
- `errorCode` - A null value or an error code as described in the [error messages list](#).
- `result` = An empty array [] or an array with details about the price, as described in the [response description](#) section.

calculatePremium

This function runs the formulas attached to the specified products or coverages - identified by the code provided in the request. The function uses other helper functions, implemented in the same library, to get the product details on a specific date, the list of items (coverages), and the formulas attached to the specified coverages. The `calculatePremium` function is called inside the endpoint only if the `isSuccess` key from the response of the `validateRequest` function is true.

Input parameters: `inputData` - The object containing the keys needed to call the endpoint.

Output parameters: An `array` containing objects with the following keys:

- `totalPremiumAmount` - The total premium amount for the product.
- `tariffType` - The **Tariff Type** defined on the product level. The option set values are: `perProduct` and `perCoverage`.

- **premiumCalculationResults** - An array with objects containing item codes and premium amounts.
- **formulaDetails** - An array with the result details - either the result returned by the formula attached on the product level OR the results returned by the formulas attached to product items (coverages), and their item codes.

Get UW Rules Result API

Use this API to get underwriting (UW) decision results, such as:

- underwriting decision results from the UW rules (formula) attached to an insurance product,
- underwriting decision results from the UW rules (formula) attached to different insurance product items (coverages).
- the UW result that was valid at a certain date, for a specified insurance product, or product item (coverage).

The **FTOS_IP_GetUWRulesResultAPI** endpoint runs the UW formulas assigned to each product, or product item (coverage), simulating the **Test Scenario** functionality - available for users in **Innovation Studio**.

NOTE

The results are in line with the **Underwriting Type** set at the product level. An insurance product can have the underwriting type set either to **perProduct** or **perCoverage**.

For a valid API request, include all the formula **Input Keys** expected by the **Business Formulas** engine, for each underwriting **Context Type** (see below). When needed, use the **Validity Date** key to get the underwriting decision result that was valid at that date, for the specified product, or product item (coverage).

HINT

All the requests through this API and their responses can be saved into the **FTOS_IP_**

ProductInterogationHistory entity. This action is available if the system parameter **FTOS_IP_ProductInterogationHistory_Enablelog** value is set to **1**.

Example

A user makes a call for the underwriting decision result for an insurance coverage, based on the code of the insurance product. The user wants the decision which was valid at a certain date - respectively, 2021-05-21.

```

1 | let p = { "productCode": "HH01", "validityDate": "2021-
   | 05-21", "sourceRecordName": "P7856", "sourceRecordId":
   | "0AFE8D0B-F423-44F4-9638-DEE696BF0B0E", "uwRulesDetails":
   | [{ "contextType": "Underwriting", "itemCode": "HA01", //
   | this key is not available for uw type = per product
   | "details": { "resistanceStructure": "Metal", "usageType":
   | "Main residence", "constructionYear": 2000, "coverage":
   | "Content", "insuredAmount": 200000 } }]
2 | }; ebs.callActionByNameAsync("FTOS_IP_
   | GetUWRulesResultAPI", p).then(function(e) { console.log
   | (e)
3 | })

```

Request data parameters

The following data parameters must be included in the request:

Parameter	Description
contextType	The context type object key needed to run the UW formula. (The context type defined for that UW formula - for example: underwriting, medical underwriting, pet underwriting etc.)
details	An object containing different keys needed to run the UW formula. The object structure is used to test the UW formula. The structure can be different for each product, or product item (coverage), based on the formula configuration.
itemCode	The item code - this key is not available for products that have <code>perProduct</code> UW type.
productCode	The code of the Insurance Product .
sourceRecordId	The Id of the record that the system is calculating the premium for.

Parameter	Description
sourceRecordName	Text identifying the record that the system is calculating the premium for.
uwRulesDetails	An array with object keys, described below.
validityDate	The reference date, prior to the current date, for getting an earlier version of the formulas structure, for the specified product. This key is not mandatory. When it is not provided, the API calls the current version of the formula. Accepted formats are: yyyy-mm-dd, dd/mm/yyyy, dd-mm-yyyy, or dd.mm.yyyy.

Response

This is an example of a response:

```

1  { "isSuccess": true, "errorMessage": null, "errorCode":
    null, "result": [ { "contextType": "Underwriting",
2  "itemCode": "HA01", // this key is not available for uw
    type = per product, "finalDecision": "Passed" // result
    of the formula "decision": { "resistanceStructure":
    "Metal", "usageType": "Main residence",
    "constructionYear": 2000.0, "coverage": "Content",
    "insuredAmount": 200000.0, "uwBuildingSumInsured":
    "Passed", "uwConstructionYear": "Passed",
    "uwResistanceStructure": "Passed", "uwUsageType":
    "Passed", "uwDecision": "Passed" } } ]
    }

```

Response description:

Key	Description
Error code	Error code.
Error message	Error message.
isSuccess	Marks whether the request was successful or not.
result	Array of objects containing details about the UW decision results.
contextType	The context type defined for that UW formula - for example: underwriting, medical underwriting, pet underwriting etc.
itemCode	The item code - this key is not available for products that have perProduct UW type.
finalDecision	Final result of the UW formula
decision	The result details returned after running the UW formula.

Error Messages

The following are the error messages that can be encountered while calling the **Get UW Rules API**:

Code	Text	Description
ERR.IP.50101	ERR.IP.50101 - Invalid validityDate format! Please use yyyy-mm-dd or dd/mm/yyyy or dd-mm-yyyy or dd.mm.yyyy!	Invalid date format for validityDate input parameter.
ERR.IP.50102	ERR.IP.50102 - Invalid validityDate !	Invalid date for validityDate input parameter.
ERR.IP.50103	ERR.IP.50103 - Invalid request!	Missing parameters in the request.
ERR.IP.50104	ERR.IP.50104 - No active product identified!	No active insurance product found.
ERR.IP.50105	ERR.IP.50105 - Error! The {code of product} insurance product was not approved at the requested date!	No active insurance product found at the date specified in validityDate input parameter.
ERR.IP.50106	ERR.IP.50106 - Product doesn't have any product items* configured!	The insurance product identified has no insurance items.
ERR.IP.50107	ERR.IP.50107 - Invalid insurance item code.	Invalid product item code.
ERR.IP.50108	ERR.IP.50108 - sourceRecordName is mandatory for premium calculation!	Source record name is mandatory.
ERR.IP.50109	ERR.IP.50109 - sourceRecordId must be uniqueidentifier type!	If transmitted, it has to be a GUID.

*Product items are the product coverages.

Endpoints

The **FTOS_IP_GetUWRulesResultAPI** endpoint runs the UW formulas attached to the specified insurance product, or product items (coverages) and returns the underwriting decision results.

Server Side Script Library

FTOS_IP_InsuranceProductAPIs

From this library, use the **GetUWRulesResult** function (described below). This function wraps all the functions necessary to validate and return the underwriting decision results.

validateRequest

This function validates the request fields.

Input parameters: **inputData** - The object containing the keys needed to call the endpoint.

Output parameters: An **object** containing the following **keys** to describe the result of the validation:

- **isSuccess** - true/ false - The boolean that shows whether the validation is successful.
- **errorMessage** - A null value or an error message as described in the [error messages list](#).
- **errorCode** - A null value or an error code as described in the [error messages list](#).
- **result** = An empty array [] or an array with details about the UW formula input parameters, as described in the [response description](#) section.

getUWRulesResult

This function executes the following:

- Runs the underwriting (UW) formulas attached to the specified product, or product items - coverages identified by their item code, provided in the request).
- Uses other helper functions, implemented in the same library, to get the details about the product, or coverages, and the attached UW formulas.

Input parameters: `inputData` - The object containing the keys needed to call the function.

Output parameters: An `array` containing objects with the following keys:

- `contextType` - The underwriting context type, set on the formula attached to the specified product.
- `itemCode` - The code of the item - this key is not available for products that have `perProduct` UW type.
- `finalDecision` - The final result of the UW formula.
- `decision` - An object that contains the keys and values from the **details input object** and also the corresponding keys and values for the decision results.

API Calls History

This functionality stores API requests and their corresponding responses. The trigger for generating records are the requests sent to the following APIs:

- [FTOS_IP_GetUWRulesResultAPI](#),
- [FTOS_IP_PremiumAmountAPI](#),
- [FTOS_IP_ProposalConfigPremiumCalculation](#).

FTOS_IP_ProductInterogationHistory

User Journey

The **FTOS_IP_ProductInterogationHistory** default form driven flow is based on the **FTOS_IP_ProductInterogationHistory** entity, that stores the API calls history. The flow is used to check and export the **FTOS_IP_ProductInterogationHistory** log.

Form

The **productInterogationHistory** form is used to view the record details.

View

The **productInterogationHistory** view is used to see the list of all the API product interrogation records registered in the system. This view can be accessed inside the **Innovation Studio Product Factory** menu, from menu item **Product Interrogation History**.

System Parameter

FTOS_IP_ProductInterogationHistory_Enablelog - This system parameter is used to set whether the details for API calls from above will be saved into **FTOS_IP_ProductInterogationHistory** entity or not. There are 2 values available for this parameter:

- 1 = the result of "proposal Configurator API", "get Prices API", "get UW Rules Result API" will be saved in "FTOS_IP_ProductInterogationHistory" entity. For "proposal Configurator API" will be saved one record for each quote included in the request.
- 0 = the result of "proposal Configurator API", "get Prices API", "get UW Rules Result API" will not be saved in "FTOS_IP_ProductInterogationHistory" entity.

NOTE

It is mandatory to set the system parameter **FTOS_IP_ProductInterogationHistory_**

Enablelog value to **1**, in order to save the requests and their corresponding responses.

Insurance Product General Operations

The **Insurance Product Factory** solution enables you to create, adjust and maintain **Insurance Products**. Read below about the endpoints, scripts and libraries used for implementing this functionality.

Server Side Library

FTOS_IP_InsuranceProduct_Operations

This server side script library stores methods for various actions that a user can perform on an insurance product, or coverage. These methods were divided in two main classes, in order to keep the functionalities on the product separate from the ones on the product items (coverages).

Class **InsuranceProduct**

This class holds methods only used for operations performed at the product level.

getProductIdOnProductItem

This function returns a list with only one product Id that was found based on the existing lookup on a specific insurance product item (coverage) Id.

Input parameters: **productItemId** - The Id of the insurance product item (coverage).

Output parameters: List with the product Id.

getProductUnderwritingType

This function returns a list with only one underwriting type that was found for a product with a specific Id.

Input parameters: `productId` - The Id of the insurance product.

Output parameters: List with the product underwriting type.

validateProductUwType

This function returns an object that will contain the underwriting type set on a specific insurance product

Input parameters: `productId` - The Id of the insurance product for which the underwriting type will be returned.

Output parameters: `validationObj` - The object containing the underwriting type.

getProductTariffType

This function returns a list with only one tariff type that was found for a product with a specific Id.

Input parameters: `productId` - The Id of the insurance product.

Output parameters: List with the product tariff type.

validateProductTariffType

This function returns an object that contains the tariff type set on a specific insurance product.

Input parameters: `productId` - The Id of the insurance product for which the tariff type is returned.

Output parameters: `validationObj` - The object containing the tariff type.

deleteProductItemFormulaByProductId

This function deletes the formulas attached on a specific insurance product.

Input parameters:

- `entityNamesList` - The list of entities.
- `insuranceProductId` - The Id of the insurance product.

Output parameters: N/A.

Class InsuranceProductItem

This class holds methods only used for operations performed at the coverage level.

getItemFormulaOnContext

This function returns a list with underwriting formulas attached on coverage level - for a specific product Id, insurance item Id and context type Id.

Input parameters:

- `insuranceProductId` - The Id of the specific insurance product item (coverage).
- `insuranceProductItemId` - The Id of the specific insurance product.
- `contextTypeId` - The Id of the specific context type.

Output parameters: List with underwriting formulas.

formulaBeforeActionValidations

This function validates if the newly added formula contains a unique combination of insurance product, insurance product item (coverage) and context type. If the data is duplicated or the formula name already exists, this function throws an error.

Input parameters:

- **coverageFormulaName** - The name of the formula which is about to be inserted or updated.
- **insuranceProductId** - The Id of the selected insurance product.
- **insuranceProductItemId** - The Id of the selected insurance product item (coverage).
- **contextTypeId** - The Id of the selected context type.
- **maxNumber** - The maximum number of records necessary for validation.

Output parameters: N/A.

ifFormulaHasChangedUpdateDataMapping

This function checks the attached underwriting formula for updates. Next, if it finds any change, the function updates the specific record from the **FTOS_CALC_DataMapping** entity.

Input parameters:

- **formulaId** - The Id of the selected formula.
- **insuranceProductItemId** - The Id of the selected insurance product item (coverage).
- **contextTypeId** - The Id of the selected context type.
- **dataMappingId** - The Id of the related data mapping record.

Output parameters: N/A.

createDataMapping

This function is called when the user clicks the **Map Data** button for inserting a new record in the **FTOS_CALC_DataMapping** entity, with some data from the underwriting formula. It also updates the related formula with the new data mapping Id.

Input parameters:

- `formulaId` - The Id of the selected formula.
- `contextTypeId` - The Id of the selected context type.

Output parameters: N/A.

insertNewDataMapping

This function inserts a new record in the **FTOS_CALC_DataMapping** entity, with some data from the underwriting formula.

Input parameters:

- `ipFormulaData` - The data object for the selected formula.
- `formulaId` - The Id of the selected formula.

Output parameters: N/A.

uwFormulaVersioningHelper

This function runs some validations before calling the **insertNewDataMapping()** function.

Input parameters:

- `insuranceProductData` (`ipFormulaData`) - The data object (containing details of the insurance product) for the selected formula.

- `insuranceProductId` - The Id of the selected insurance product item (coverage).
- `contextTypeId` - The Id of the selected context type.
- `formulaId` - The Id of the selected formula.

Output parameters: N/A.

getInsuranceProductItemDetails

This function returns a list with some data objects for the insurance coverages found.

Input parameters: `productId` - The Id of the selected insurance product.

Output parameters: List with insurance coverages.

getInsuranceProductItemDetails

This function returns different attributes from the **FTOS_IP_InsuranceProductItem** entity, based on the product id.

Input parameters: `productId` - The Id of the selected insurance product.

Output parameters: `query` - An array that contains an object with the following results: `FTOSIPInsuranceProductId`, and `Name`.

getInsuranceProductItemDetailsVersioned

This function returns different attributes from a versioned record of the **FTOS_IP_InsuranceProductItem** entity, based on the versioned product id;

Input parameters: `productId` - The Id of the selected insurance product.

Output parameters: `query` - An array that contains an object with the following results: `Percentage`.

getProductDetails

This function returns different attributes from a record of the **FTOS_IP_InsuranceProduct** entity, based on the product id.

Input parameters: `productId` - The Id of the selected insurance product.

Output parameters: `query` - An array that contains an object with the following results: `attributeVersion`, `Name`, `TariffType`, `ReferencedAttributeId`, and `ClonedProductId`.

getCoverageSplitDetails

This function returns different attributes from a record of the **FTOS_IP_PremiumCoverageSplit** entity, based on the product coverage id and product formula id.

Input parameters:

- `productItemId` - The Id of the product coverage.
- `productFormulaId` - The Id of the product formula.

Output parameters: `query` - An array that contains an object with the following results: `Name`.

getSplitDetails

This function returns different attributes from a record of the **FTOS_IP_PremiumCoverageSplit** entity, based on the product coverage id.

Input parameters: `coverageId` - The Id of the product coverage.

Output parameters: `query` - An array that contains an object with the following results: `Name`, and `Percentage`.

getProductFormula

This function returns different attributes from a record of the **FTOS_IP_InsuranceProductFormula** entity, based on the product id.

Input parameters: `productId` - The Id of the insurance product.

Output parameters:

`query` - An array that contains an object with the following results:
`Name`.

getProductDetailsByCoverage

This function returns information about a product, based on the coverage Id.

Input parameters: `parentId` - The Id of the coverage.

Output parameters: The query object.

getProductDetailsByModule

This function returns information about a product, based on the module Id.

Input parameters: `moduleId` - The Id of the module.

Output parameters: The query object.

getProductDetailsByFormula

This function returns information about a product, based on the product formula Id.

Input parameters: `formulaId` - The Id of the product formula.

Output parameters: The query object.

getProductDetailsByItemFormula

This function returns information about a product, based on the coverage formula Id.

Input parameters: `formulaId` - The Id of the coverage formula.

Output parameters: The query object.

getProductDetailsByFormulaUnd

This function returns information about a product, based on the product underwriting formula Id.

Input parameters: `formulaId` - The Id of the product underwriting formula.

Output parameters: The query object.

getProductDetailsByItemFormulaUnd

This function returns information about a product, based on the coverage underwriting formula Id.

Input parameters: `formulaId` - The Id of the coverage underwriting formula.

Output parameters: The query object.

getFormulaDataMapping

This function returns information about a formula data mapping, based on the data mapping Id.

Input parameters: `mappingId` - The Id of the data mapping.

Output parameters: The query object.

checkStatus

This function checks if the status of a product is in Draft or Version draft status and throws an error.

Input parameters:

- **id** - The Id used to retrieve the product's information.
- **functionName** - The name of the function used to retrieve the product's information.
- **resource** - The message the error throws when the product is not in **Draft** or **Version Draft** status.

Output parameters: N/A.

Endpoints

FTOS_IP_CheckBusinessStatus

Endpoint used to run the **FTOS_IP_CheckBusinessStatus** server automation script.

FTOS_IP_CheckBusinessStatusItem

Endpoint used to run the **FTOS_IP_CheckBusinessStatusItem** server automation script.

FTOS_IP_CheckBusinessStatusModule

Endpoint used to run the **FTOS_IP_CheckBusinessStatusModule** server automation script.

FTOS_IP_CheckBusinessStatusByFormula

Endpoint used to run the **FTOS_IP_CheckBusinessStatusByFormula** server automation script.

Server Side On-demand Scripts

FTOS_IP_CheckBusinessStatus

Based on the provided insurance product ID, this script checks if the product is in draft or version draft status and returns a boolean.

Input parameters: `productId` - The insurance product Id.

Output parameters: `check` - The boolean that determines if the product is in the correct status.

FTOS_IP_CheckBusinessStatusItem

Based on the provided insurance product item ID, this script checks if the product is in draft or version draft status and returns a boolean.

Input parameters: `itemId` - The insurance product item Id.

Output parameters: `check` - The boolean that determines if the product is in the correct status.

FTOS_IP_CheckBusinessStatusModule

Based on the provided insurance product module ID, this script checks if the product is in draft or version draft status and returns a boolean.

Input Parameters: `moduleId` - The insurance product module Id.

Output parameters: `check` - The boolean that determines if the product is in the correct status.

FTOS_IP_CheckBusinessStatusByFormula

Based on the provided insurance product formula ID, this script checks if the product is in **Draft** or **Version Draft** status and returns a boolean.

Input parameters: `formulaId` - The insurance product formula Id.

Output parameters: `check` - The boolean that determines if the product is in the correct status.

Cloning Insurance Products

This functionality allows the user to generate a new insurance product based on an existing one. The cloned product captures all the available information from the original product. The user must select an existing product and press the **Clone** button available on the product view. This action triggers the opening of a specific pop-up form where the user can insert a new name and a new code for the new product. Upon saving, the clone product is generated, in **Draft** status.

Read below about the endpoints, scripts and libraries used for implementing this functionality.

Endpoints

The following endpoints are used to perform the cloning process:

FTOS_IP_ValidateIP

This endpoint receives an object as a parameter. The object contains the product name and code inserted in the cloning pop-up form, connected to the original product. The endpoint takes those values and validates them, to make sure there is no other product with the same name and code.

FTOS_IP_CloneIP

This endpoint receives an object as a parameter. The object contains the new product name and code, obtained from the original product form. The script linked to this endpoint takes these values and inserts a cloned product in the system. The clone product contains the same data, similar to the original

one.

Server Side Script Libraries

FTOS_IP_InsuranceProduct_Clone

This library is used to clone insurance products. From this library, the following functions are used:

GetData

This object wraps smaller functions that fetch data from different entities.

getProductInfo

This function fetches the **Name** and the **Code** of an insurance product based on a given name or code.

Input parameters:

- **entityName** - The name of the entity on which the fetch will be made.
- **ipName** - The name of the new insurance product.
- **ipCode** - The code of the new insurance product.

Output parameters: A list of insurance products, found by name or code.

getEntityId

This function fetches the **Id** of an entity based on a given name from the entity metadata.

Input parameters: **entityName** - The name of the entity needed.

Output parameters: The **Id** of the entity, found by the fetch.

checkEntityInVersionSettings

From the **FTOS_VersionSettings** entity, this function fetches all the records related to the entity that is about to be cloned.

Input parameters: **selectedEntity** - The Id of the entity from which the record is about to be cloned.

Output parameters: A list of all the related records.

getRelatedEntities

From the **FTOS_VersionSettingsItem** entity, this function fetches all the items related to the entity that is about to be cloned.

NOTE

Parts of the product are stored in different entities, for more details see the [Importing Insurance Products](#) page.

Input parameters: **versionSettingsid** - The Id of the entity that is about to be cloned.

Output parameters: A list of all the related items.

getAlias

This function subtracts the first letter from an entity name to use it as an alias.

Input parameters: **entityName** - The name of the entity.

Output parameters: The **first letter** from the entity name.

getEntityData

This function fetches the data from an entity. The fetch uses the entity name received as input and it searches according to the conditions written by two other parameters: the attribute name and the value it should have.

Input parameters:

- `entityName` - The name of the entity on which the fetch is performed.
- `conditionAttrName` - The name of the attribute found inside the condition.
- `conditionRecord` - The value of the attribute found inside the condition.

Output parameters: A list of all the records returned by the fetch.

Validations

This object wraps smaller functions that validate different sets of data.

isNullOrEmpty

This function checks if the value received through a parameter is valid. If the value is valid, the function returns true. Otherwise, it returns false.

Input parameters: `value` - The value which is about to be validated.

Output parameters: **Boolean** value (true/ false).

validateNameCode

From the client side, this function receives the values inserted into the fields of the cloning pop-up - namely, the values for **Name** and **Insurance Product Code**. If the values come in as being empty or invalid, an exception will be thrown to inform the user. The same thing happens if the values are already used by the system (an existing insurance product has the same name or the product code).

Input parameters:

- `entityName` - The name of the entity on which the fetch will be made.
- `ipName` - The name of the new insurance product.
- `ipCode` - The code of the new insurance product.

Output parameters: If no exception is thrown, the function quits and the cloning process moves on.

ProductClone

This object wraps functions that perform various steps of the cloning process.

cloneRelatedEntities

This function receives a list of **Version Setting Items** as a parameter and loops through it for cloning each one of the items into the new product.

Input parameters:

- `versionSettingsItems` - The list of the version setting items.
- `productId` - The Id of the product which is about to be cloned.
- `newCloneId` - The Id of the new product that receives all the data from the original one.

Output parameters: N/A.

insertClonedProduct

This function inserts into an entity an object with data, both received as a parameter, after some validations are being made on a few attribute names.

Input parameters:

entityName - The name of the entity on which the insert will be made.

entityData - The object containing all the data from the original product.

productId - The id of the cloned product.

Output parameters: Returns an Id of the newly inserted record.

cloneRelatedRecord

This function is similar with the function above. The function inserts into an entity an object with data, both received as a parameter, after some validations are being made on a few attribute names.

Input parameters:

- **entityName** - The name of the entity on which the insert is made.
- **entityData** - The object containing all the data from the original product.
- **recordId** - The Id of the product which is about to be cloned.
- **relatedAttr** - The name of an attribute that get as value the **recordId** parameter.

Output parameters: Returns an id of the newly inserted record

getChildEntities

This function the function firstly fetches from the entity metadata a record based on an name received through the parameter. After that, another fetch is made on the **FTOS_VersionSettingsItem** entity using the Id obtained from the fetch before as a condition for the **parentVersionedEntity** attribute.

Input parameters: **entityName** - The name of the entity on which the fetch is made.

Output parameters: Returns a list of all the records found by the second fetch.

getPrimaryKeyValue

This function returns the value of a record's primary key from an entity with a specified name.

Input parameters:

- `entityName` - The name of the entity from which the primary key is needed.
- `record` - The record from the entity.

Output parameters: The **primary key** value.

clone

This function clones all the related entities of a product along with their "children" entities.

NOTE

Parts of the product are stored in different entities, for more details see the [Importing Insurance Products](#) page.

Input parameters:

- `entityName` - The name of the entity on which the fetch is made.
- `recordToBeClonedId` - The Id of the record which is about to be cloned.
- `recordToBeClonedPrimaryAttribute` - The Id of the record's primary attribute.
- `clonedRecordId` - The Id of the new record.

Output parameters: N/A

cloneProductData

This function calls all the other functions (detailed above) for validating, inserting and updating all the data for the product to be cloned.

Input parameters:

- `entityName` - The name of the entity on which the fetches are made.
- `productId` - The Id of the product that is about to be cloned.
- `ipName` - The name of the new insurance product.
- `ipCode` - The code of the new insurance product.

Output parameters: N/A

getEntityAttributes

This function returns all the attributes from an entity.

Input parameters: `entityName` - The name of the entity on which the fetches are made.

Output parameters: `query` - The resulting array, with all the attributes.

Importing Insurance Products

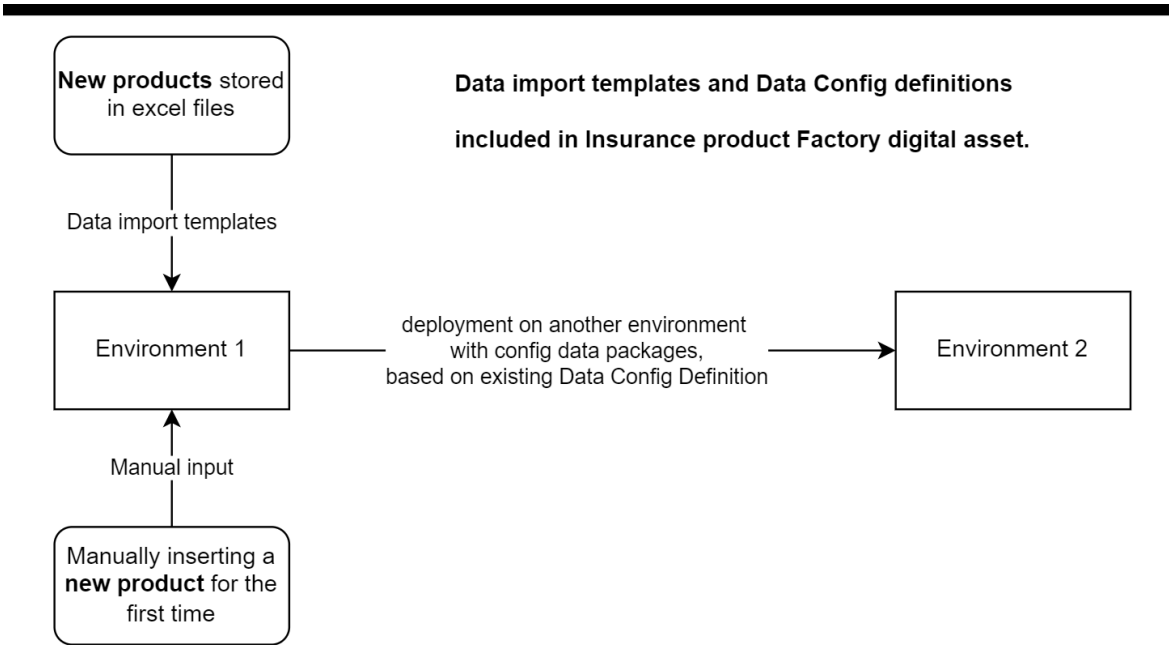
check links before release

The solution's **Data Import Templates** allow you to add multiple types of product data on a **FintechOS** destination environment. The most important use case for this functionality is a fast bulk import into the system of numerous new insurance products - with all imported products being registered in **Draft** status. But this is not

the only use case. For example, using the **Data Import Templates**, a user may import only some data about specific product components - say only tariff and underwriting formulas for a certain product. Consequently, templates are different depending on what kind of data needs to be added.

NOTE
 A newly imported product is registered in **Draft** status, meaning it is editable. After import, use **Innovation Studio** to complete your product configurations and also, to attach premium calculation and underwriting **formulas** to it. Once finished, approve your product in order for it to go live.

The following is an example of use cases for importing insurance products:



See below the **Data Config Definitions** and **Data Import Templates** available for the **Insurance Product Factory** solution.

Data Import Templates

Use [Data Import Templates](#) to import a product for the first time on a destination environment. However, for deploying changes on existing products, use [Data Config Definition](#), as it has the versioning mechanism embedded.

HINT

For downloading a specific example template file listed below, click the corresponding link. For downloading all the files, click this [DataConfigImport Archive](#) - which includes also the **DataConfigImport.xml** file.

The following are the **Data Import Templates** available for the **Insurance Product Factory** solution:

Product Templates

- [FTOS_IP_InsuranceType](#) - This template will import a new **Insurance Type**.
- [FTOS_IP_InsuranceProduct](#) - This template will import a new **Insurance Product**.
- [FTOS_IP_InsuranceItem](#) - This template will import a new **Insurance Coverage Type**. *Insurance Item Type was the name of the entity. But considering it was replaced with the LOB subtype, we will probably update this page in the near future*
- [FTOS_IP_InsuranceProductItem](#) - This template will import a new **Coverage** on an **Insurance Product**.
- [FTOS_IP_InsuranceProductItemModule](#) - This template will import a new **Module** on a **Coverage**.
- [FTOS_IP_InsuranceRisk](#) - This template will import a new **Insurance Peril**.
- [FTOS_IP_CoveredRisk](#) - This template will import a new **Covered Peril**.
- [FTOS_IP_InsuranceCoveredRiskNew](#) - This template will import a new **Insurance Covered Peril** on a **Module**.

IMPORTANT!

After importing the files, additional checks or changes are need on the product level, as not all the attributes from the entities are available in the excel files. Missing information needs to be filled in manually before approving the product.

Product Formula Templates

- [FTOS_IP_InsuranceProductFormula](#) - This template will import a new insurance product formula.
- [FTOS_IP_InsuranceProductItemFormula](#) - This template will import a new insurance product coverage formula.
- [FTOS_IP_InsuranceProductItemFormulaUnd](#) - This template will import a new insurance product underwriting formula.
- [FTOS_IP_InsuranceProductCoverFormulaUnd](#) - This template will import a new insurance product coverage underwriting formula.
- [FTOS_CALC_DataMapping](#) - This template will import a new data mapping.
- [FTOS_IP_FormulaDataMapping](#) - This template will import a new formula data mapping.
- [FTOS_IP_PremiumCoverageSplit](#) - This template will import a new premium coverage split.

Insured Object Type Templates

[FTOS_IP_InsuredObjectType](#) - This template will import a new insured object type.

[FTOS_IP_InsuredObjectType.xlsx](#)

[FTOS_IP_InsuredObjectTypeDimension](#) - This template will import a new insured object type dimension.

[FTOS_IP_InsuredObjectTypeDimension.xlsx](#)

Data Config Definition

The **FintechOSData Config Definition** functionality allows the deployment of products to a destination environment, using the versioning settings. The same versioning rules running for the manual versioning process will run when importing the data with Data Config Definition as well.

When your Product is in **Active** status, changing it may be done only by adding a **new version**. When you choose to import the product changes in a **Data Config Definition** file, this functionality has the versioning mechanism embedded. This way you can be sure that your changes are deployed to the specified product.

IMPORTANT!

The **Data Config Definitions** needed to export a product are available in the **Insurance Insurance Product Factory Import digital asset**. Examples are also attached on the **Data Import** files step of the digital asset. Not all the entities that are part of a product definition are included in the Product's **Data Config Definition**. For some entities, the data needs to be deployed using **Data Import Templates**.

The following deployment options are available:

FTOS_IP_InsuranceType

Use data import template to deploy the records.

FTOS_IP_InsuranceltemType

Use data import template to deploy the records.

FTOS_IP_InsuranceRisk

Data config definition available, exporting data from **FTOS_IP_InsuranceRisk**.

FTOS_IP_InsuredObjectType

Use data import template to deploy the records.

FTOS_IP_UnderwritingContextType

Use data import templates to deploy the records.

FTOS_IP_InsuranceProduct

- Data config definition available, exporting data from **FTOS_IP_InsuranceProduct**, **FTOS_IP_InsuranceProductItem** (coverages), **FTOS_IP_InsuranceProductItem** (Modules).
- **is Versionable flag enabled**
- **Premium Amount Formulas** and **Underwriting Formulas** will be exported as well. Make sure the **Formulas** are already available on the destination environment when you deploy the product.

NOTE

The Insurance Product Factory **Import Formulas Digital Asset** contains **Formulas** used by the demo **Product** included in the Insurance Product Factory **Import Digital Asset**.

IMPORTANT!

Links with **FTOS_IP_InsuranceType**, **FTOS_IP_InsuranceItem**, **FTOS_IP_InsuranceItem**, **FTOS_IP_InsuranceItem**, **FTOS_IP_InsuranceItem**, **FTOS_IP_InsuranceItem** records will be made, so the needed records from these entities **must be imported first!**

FTOS_IP_CoveredRisk

Data config definition available, exporting records from **FTOS_IP_CoveredRisk** entity and creating a link to **FTOS_IP_InsuranceRisk** records, previously imported.

FTOS_IP_InsuranceCoveredRisk

Use data import template to deploy the records.

Below, an example of the **Order Index** for importing **Templates** and **Definitions**:

<input type="checkbox"/> Data Import Template	Data Config Definition	Type	Order Index	File
Q	Q	Q	Q	Q
FTOS_IP_InsuranceType		Based On Data Import Templates	1	Download
FTOS_IP_InsuranceItemType		Based On Data Import Templates	2	Download
	FTOS_IP_InsuranceRisk	Based On Data Config Definition	3	
FTOS_IP_UnderwritingContextType		Based On Data Import Templates	4	Download
	FTOS_IP_InsuranceProduct	Based On Data Config Definition	5	
	FTOS_IP_CoveredRisk	Based On Data Config Definition	6	
FTOS_IP_InsuranceCoveredRiskNew		Based On Data Import Templates	7	Download

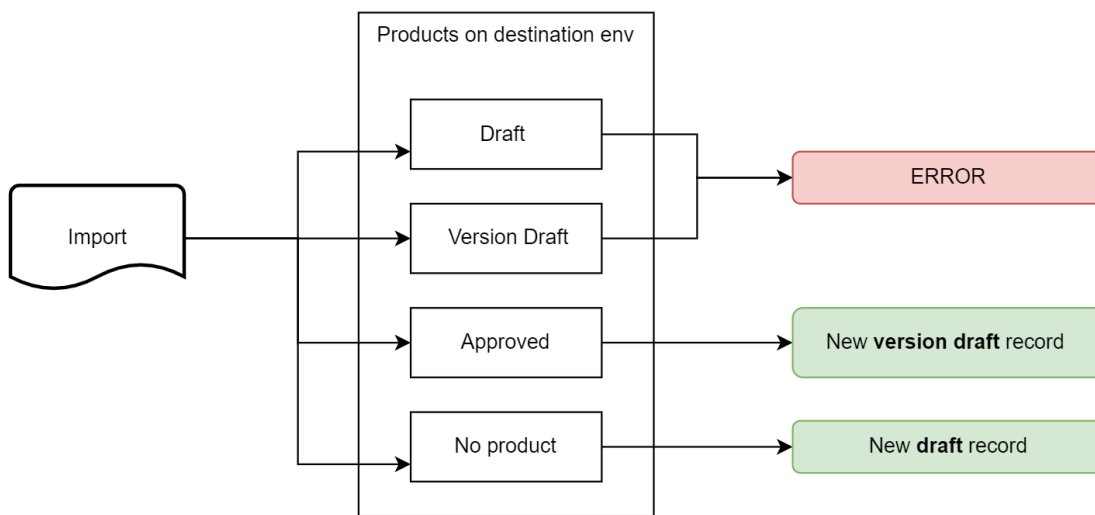
Troubleshooting Aspects

IMPORTANT!

Since the definition used to migrate records from the **FTOS_IP_InsuranceProduct** has the **is versionable** flag enabled, and since the versioning mechanism is available on products, **versioning rules will be applying for each deployment.**

- If, on the destination environment, the **Product is NOT available** (identified by the unique constraint set on **FTOS_IP_InsuranceProduct** entity), then a new product with **Draft** status is added.
- If, on the destination environment, the **Product is already available** (identified by the unique constraint set on **FTOS_IP_InsuranceProduct** entity) and its status is **Draft**, then the import is not possible, and an error is triggered.
- If, on the destination environment, the **Product is already available** (identified by the unique constraint set on **FTOS_IP_InsuranceProduct** entity), and its status is **Approved** (with **NO Version Draft** records), then a new Product version (with **Version Draft** status) is created.
- If, on the destination environment, the **Product is already available** (identified by the unique constraint set on **FTOS_IP_InsuranceProduct** entity), and its status is **Approved** (with **Version Draft** records), then the import is not possible, and an error is triggered.

Below, an example of results obtained by importing between environments:



HINT
 For more details, consult also the [Data Import Templates](#) the [Data Config Definition](#), and the [Configuration Data Package](#) documentation.

Configuring Payment Types

This functionality allows the user to configure types of payment that are going to be available for an insurance product, and also select a **Main Payment Type** for that product. The payment type form is available inside the [Main Info](#) tab of the product, in the **Payments Schedule & Billing** section.

Read below about the entity, flows and views used for implementing this functionality.

FTOS_IP_PaymentType

This entity stores data about the payment types available for each product.

Default - Form Driven Flow

The **Default** form driven flow is based on the **FTOS_IP_PaymentType** entity. The flow is used for inserting, updating or deleting the payment types for an insurance product.

Default - View

The **Default** view is used to see the list of all the payment types, available for a product. This view can be accessed inside the **Innovation Studio Product** menu, in the **Main Info** tab.

Configuring Insured Object Types

General process description: Inside the **Insured Object Types** section, the user can create a new object, and configure its dimensions. Inside the dimensions grid, the user can change the order index of the grid items.

NOTE

The **addressId** dimension is added by default, for every object.

Read below about the entities, endpoints, scripts and libraries used for implementing this functionality.

Entity FTOS_IP_InsuredObjectType

This entity stores data about the insured object types.

Form Driven Flows

Default

This flow is used for creating insured object types.

FTOS_IP_InsuredObjectType

Insured Object Type step: This flow is used to edit insured object types. It also allows the user to configure the dimensions for the insured object type.

History step: This displays the versioning history of the record.

Views

Default

This view displays all registered insured object types - in **Draft**, **Approved** or **Closed** status.

FTOS_IP_InsuredObjectType_History

This view displays the version history for the specified record.

FTOS_IP_InsuredObjectType_ByBusinessStatus

This view displays all the **Approved** insured object types.

Entity FTOS_IP_InsuredObjectTypeDimension

This entity stores data about the insured object types dimensions.

Form Driven Flows

Default

This flow is used for creating and editing dimensions on insured object types.

Views

Default

This view displays all the dimensions defined on an insured object type. Inside the dimensions grid, the user can change the order index of the grid items. This also sets the order index for how dimensions are displayed further, on the policy form.

DimensionsByInsuredObjectType

This view displays all available insured object types dimensions on an `insuredObjectId` received as parameter.

Entity Attribute

A new view was developed on the **FintechOS Attribute** entity. The **filterByEntityId** view displays all the attributes from a specific entity, based on the entity's Id.

HINT

For more details - such as an entity's Data Model, Data Events, Data Referencing, Data API, Unique Constrains, and more, go to **Innovation Studio > Data Model Explorer > Business Entities List** and check the entity's description. For an overview about how **FTOS entities** are described, consult the [Data Model Explorer](#) documentation.

Endpoints

FTOS_IP_getEntityFromObjectType

Endpoint used to run the **FTOS_IP_getEntityFromObjectType** server automation script.

FTOS_IP_InsuredObjectType_ CheckBusinessStatus

Endpoint used to run the **FTOS_IP_InsuredObjectType_ CheckBusinessStatus** server automation script.

FTOS_IP_CloneObjectType

Endpoint used to run the **FTOS_IP_CloneObjectType** server automation script.

FTOS_IP_ValidateObjectName

Endpoint used to run the **FTOS_IP_ValidateObjectName** server automation script.

FTOS_IP_InsuredObjectType_GetDetails

Endpoint used to run the **FTOS_IP_InsuredObjectType_GetDetails** server automation script.

Server Side On-demand Scripts

FTOS_IP_getEntityFromObjectType

Based on the Id of the insured object type, this script returns the Id of the mapped entity. This script displays only the mapped entity's attributes when configuring the dimensions.

Input parameters: `objectId` - The Id of the insured object type.

Output parameters: `entityMapping` - The Id of the mapped entity.

FTOS_IP_InsuredObjectType_ CheckBusinessStatus

Based on the provided insured object type Id, this script checks if the insured object type is in **Draft** or **Version Draft** status and returns a boolean.

Input parameters: `insuredObjectType` - The Id of the insured object type.

Output parameters: `check` - The boolean that determines if the insured object type is in the correct status.

FTOS_IP_CloneObjectType

Based on the insured object type's name and the name of the entity, this script starts the cloning process for the **FTOS_IP_InsuredObjectType** entity.

Input parameters:

- `objectName` - The name of the insured object type.
- `entityName` - The name of the entity.

Output Parameters: N/A

FTOS_IP_ValidateObjectName

This script checks if the name provided by the user for the new insured object type (for the cloning process) is valid.

Input parameters: `objectName` - The name proposed for the insured object type.

Output Parameters: `objectFound` - The boolean that determines if there is already an object with the provided name.

FTOS_IP_InsuredObjectType_GetDetails

Based on the Id of the insured object type, this script returns the details of the object type: attribute version, description, the Id of the mapped entity.

Input parameters: `objectTypeId` - The Id of the insured object type.

Output Parameters: `objectTypeDetails` - An object containing the insured object type's details.

Server Side Libraries

1. FTOS_IP_InsuredObjectType_Operations

This server side script library stores methods for various actions that a user can perform on an insured object type. These methods were divided in two main **classes** to keep the functionalities on the insured object type separate from the ones on the insured object type dimension.

class InsuredObjectType

This class holds methods used only for operations performed at the insured object type level.

getObjectTypetDetailsById

This function returns information about an insured object type, based on the insured object type Id

Input parameters: **objectId** - The Id of the insured object type.

Output parameters: **query object** - An object containing the results of the query.

checkStatus

This function checks if the status of an insured object type is in Draft or Version draft status and throws an error.

Input parameters:

- **id** - The **Id** used to retrieve the product's information.
- **functionName** - The name of the function used to retrieve the product's information.
- **resource** - The message the error throws when the product is not in **Draft** or **Version Draft** status.

Output parameters: N/A.

checkIfUsedByApprovedProduct

Based on the object type's id, this function checks if the object type is used by an approved product or not.

Input parameters: `objectTypeId` - The Id of the object type.

Output parameters: N/A.

class InsuredObjectTypeDimension

This class holds methods used only for operations performed on a insured object type dimension level. From this class, the **getDimensionDetailsByObject** function is used.

This function returns information about an insured object type dimension, based on the insured object type Id.

Input parameters: `objectTypeId` - The Id of the insured object type.

Output parameters: **query object** - An object containing the results of the query.

2. FTOS_IP_InsuredObjectType_Clone

This server side script library stores a **class** of methods that deal with cloning an insured object type.

class InsuredObjectType

getObjectInfo

This function returns information about an insured object type, based on its name.

Input parameters: `objectName` - The name of the insured object type.

Output parameters: **query object** - An object containing the results of the query.

validateName

This function receives the values of the **Name** field inside the cloning pop-up, from the client side. If the value comes in as being empty or invalid, an exception will be thrown to inform the user. Same thing happens if the name points to an existing insured object type with the same values.

Input parameters: `objectName` - The name proposed for the insured object type to be cloned.

Output parameters: If no exception gets thrown, the cloning process moves on.

insertClonedObject

After performing some validations on a few attribute names, this function inserts a data **object** into an **entity**, both received as a parameter.

Input parameters:

- `entityName` - The name of the entity on which the insert will be made.
- `entityData` - The object containing all the data from the original insured object type.
- `objectId` - The Id of the cloned insured object type.

Output parameters: Returns an `Id` of the newly inserted record.

cloneObjectData

This function calls all the other functions (described above) in order to validate, insert and update all the data for the new clone.

Input parameters:

- **entityName** - The name of the entity on which the fetches will be made.
- **objectId** - Id of the insured object type which is about to be cloned.
- **objectName** - The name of the cloned insured object type.

Output parameters: N/A.

Configuring Lines Of Business

The **FTOS_IP_LineOfBusiness** functionality allows the **Insurance Product Factory** users to configure **Lines of Business** specific for their insurance activity, with the following levels of granularity:

- Class of Business;
- Category of Business;
- Line of Business;
- Line of Business Subtype.

Read below about the form driven flows, scripts and endpoints used for implementing this functionality.

Form Driven Flows

Custom: FTOS_IP_LineOfBusiness

General description: The **FTOS_IP_LineOfBusiness** custom flow is available in **Innovation Studio** to support the line of businesses configuration logic. This custom flow has sections dedicated to each configurable type - providing a

view for each level that can be defined.

View Custom From the view, the user can add a new record for the corresponding level (see below), edit a record, or export the existing records.

FTOS_IP_ClassOfBusiness

General description: The **FTOS_IP_ClassOfBusiness** form driven flow is based on the **FTOS_IP_ClassOfBusiness** entity. The flow is used for inserting or editing a business class. The fields **Name** and **Display Name** are mandatory to be completed.

View Default This view is used to see the list of all the defined business classes. From this view, the user can:

- go to **FTOS_IP_ClassOfBusiness** form to edit the record - by double clicking a record;
- go to **FTOS_IP_ClassOfBusiness** form to insert a new record - by clicking the **Insert** button;
- export the list of business classes - by clicking the **Export** button.
- delete a record - by clicking the **Delete** button.

FTOS_IP_CategoryOfBusiness

General description: The **FTOS_IP_CategoryOfBusiness** form driven flow is based on the **FTOS_IP_CategoryOfBusiness** entity. This flow it is used for inserting or editing a business category . The fields **Name**, **Display Name** and **Class of business** are mandatory to be completed.

View Default This view is used to see the list of all the defined business categories. From this view, the user can:

- go to **FTOS_IP_CategoryOfBusiness** form to edit the record - by double clicking a record;
- go to **FTOS_IP_CategoryOfBusiness** form to insert a new record - by clicking the **Insert** button;
- export the list of business classes- by clicking the **Export** button.

FTOS_IP_LineOfBusiness

General description: The **FTOS_IP_LineOfBusiness** form driven flow is based on the **FTOS_IP_LineOfBusiness** entity. This flow it is used for inserting or editing a business line. The fields **Name**, **Display Name** and **Category of business** are mandatory to be completed.

View Default_v1 This view is used to see the list of all the defined business lines. From this view, the user can:

- go to **FTOS_IP_LineOfBusiness** form to edit the record - by double clicking a record;
- go to **FTOS_IP_LineOfBusiness** form to insert a new record - by clicking the **Insert** button;
- export the list of business classes- by clicking the **Export** button.

FTOS_IP_LineOfBusinessSubtype

General description: The **FTOS_IP_LineOfBusinessSubtype** form driven flow is based on the **FTOS_IP_LineOfBusinessSubtype** entity. This flow it is used for inserting or editing a subtype of a business line. The fields **Name**, **Display Name** and **Line of business** are mandatory to be completed.

View Default This view is used to see the list of all the defined business lines subtypes. From this view, the user can:

- go to **FTOS_IP_LineOfBusinessSubtype** form to edit the record - by double clicking a record;
- go to **FTOS_IP_LineOfBusinessSubtype** form to insert a new record - by clicking the **Insert** button;
- export the list of business classes- by clicking the **Export** button.

On Demand Script

FTOS_IP_GetLineOfBusinessVA

On demand script that is triggered when a **line of business** is selected on **FTOS_IP_LineOfBusinessSubtype** form or a **class of business** is selected on **FTOS_IP_LineOfBusiness** form in order to complete the **category** of business and class of business in **FTOS_IP_LineOfBusinessSubtype** form or **class** of business in **FTOS_IP_LineOfBusiness** form.

Input parameters: An object containing the `lineOfBusinessId` and `categoryOfBusinessId`.

Output parameters: An object containing `categoryOfBusinessId` and `classOfBusinessId`.

Endpoint

FTOS_IP_GetLineOfBusinessVA

This endpoint calls the **FTOS_IP_GetLineOfBusinessVA** on demand script to retrieve the **class** of business, or class of business and **category** of business. It is used on **FTOS_IP_LineOfBusiness** form and **FTOS_IP_LineOfBusinessSubtype** form.

Insurance Formulas

FintechOS Business Formulas enable you to reduce completion time for different types of calculations. Once you created your own insurance formulas, you can make them work in conjunction with your insurance products, or product items (coverages).

In the **Insurance Product Factory** solution, different entities store different formulas, as follows:

Entities For Product Formulas

The following entities store formulas attached at the level of the insurance product:

- **FTOS_IP_InsuranceProductFormula** - stores the premium calculation formulas attached to a product.
- **FTOS_IP_InsuranceProductItemFormulaUnd** - stores the underwriting formulas attached to a product.

Entities For Product Items Formulas

The following entities store formulas attached to an insurance product item (coverage):

- **FTOS_IP_InsuranceProductItemFormula** - stores the premium calculation formulas attached to a product item (coverage).
- **FTOS_IP_InsuranceProductCoverFormulaUnd** - stores the underwriting formulas attached to a product item (coverage).

HINT

For details about business use cases, consult the Insurance Business Formulas page.

Check the following pages for technical details about:

- [Attaching Formulas](#)
- [Defining Premium Splits](#)
- [Mapping Formulas](#)
- [Testing Formulas](#)
- [Demo Formulas](#)

NOTE

Only one formula can be attached to a specified product, or product item (coverage), at a time.

Attaching Formulas

Business Formulas can be attached to specific targets (product, or coverage) in order to process the inputs and generate different types of outputs - such as premium calculation or underwriting scoring.

The **Insurance Product Factory** allows the user to attach the following types of formulas:

- formulas for the premium calculation per product;
- formulas for the premium calculation per coverage;
- formulas for underwriting per product;
- formulas for underwriting per coverage.

NOTE

Only one formula can be attached on a specified product, or coverage, at a time. This condition applies to both tariff and underwriting formulas.

Read below about the flows, libraries, scripts and endpoints used by this functionality. (Scroll down or [click for attaching underwriting formulas.](#))

Attach Premium Calculation Formulas

This functionality allows the user to attach a premium calculation formula to a specified insurance product, or coverage, based on a previously indicated **Tariff Type**.

The **FTOS_IP_CalcType** option set is available for configuring the tariff type; the option set values being: **Per Product** and **Per coverage**:

- For products with **Per Product** tariff type, a premium calculation formula can be attached on the product.
When choosing this option set value, the **FTOS_IP_InsuranceProductFormula** default data form becomes available (in the **Premium Amount** tab) for inserting a premium calculation formula, or updating the existing configuration (namely, delete a previously attached formula and inserting a new one).

- For products with **Per coverage** tariff type, a premium calculation formula can be attached on the product item (coverage).
When choosing this option set value, the **FTOS_IP_InsuranceProductItemFormula** default data form becomes available (in the **Premium Amount** tab) for inserting a premium calculation formula, or updating the existing configuration (namely, delete previously attached formulas and inserting new ones).

NOTE

When changing the **Tariff Type** value, the user is warned about the automatic removal of any tariff formulas attached previously.

User Journeys

FTOS_IP_InsuranceProductFormula

General description: This default form driven flow is based on the **FTOS_IP_InsuranceProductFormula** entity. The flow is used for inserting or updating premium formulas attached on the product level. Only one formula can be attached per product. For this condition, a validation is in place - with the following error message: “There is another Insurance Product Formula record with the same Insurance Product”. The product name is pre-filled and read only and the formula is mandatory to be completed.

On edit mode, the **Map Data** button is visible. When pressing the **Map Data** button, the data mapping standard screen appears.

View Default This view lists all the premium formulas attached on the product level - showing the name, the insurance product and the formula, sorted **Ascending by Name**.

FTOS_IP_InsuranceProductItemFormula

General description: This default form driven flow is based on the **FTOS_IP_InsuranceProductItemFormula** entity. The flow is used for inserting or updating premium formulas attached on coverage level. For products with tariff type set to **Per Coverage**, a user can add a single formula for **each coverage** of the product. When inserting a new formula, the product name is

pre-filled and read-only and the insurance coverage and formula fields are mandatory to be completed.

On edit mode, the **Map Data** button is visible. When pressing the **Map Data** button, the data mapping standard screen appears.

View Default This view lists all the premium formulas attached per coverage - showing the name, the insurance coverage, the formula and order, sorted **Ascending by Order**.

On Demand Scripts

FTOS_IP_InsuranceProductFormula_ CreateDataMapping

This script inserts a new record in the **FTOS_CALC_DataMapping** with some data received for the premium formula. More than that, the script also updates the formula (**FTOS_IP_InsuranceProductFormula**) with the Id of the newly inserted data mapping calculation.

FTOS_IP_InsuranceProductItemFormula_ CreateDataMapping

This script inserts a new record in the **FTOS_CALC_DataMapping** with some data received for the premium formula. More than that, the script also updates the formula (**FTOS_IP_InsuranceProductItemFormula**) with the Id of the newly inserted data mapping calculation.

Endpoints

FTOS_IP_InsuranceProductFormula_ CreateDataMapping

It calls the **FTOS_IP_InsuranceProductFormula_CreateDataMapping** on demand script. (See above.)

FTOS_IP_InsuranceProductItemFormula_ CreateDataMapping

It calls the **FTOS_IP_InsuranceProductItemFormula_CreateDataMapping** on demand script. (See above.)

Attach Insurance Underwriting Formulas

Based on the **Underwriting Type** selected in the **Main Info** step on an insurance product, the user can attach underwriting formulas, either on product or coverages level. Depending on that selection, a grid of records showing the formulas attached is displayed on the **Underwriting** tab.

NOTE

When changing the underwriting type value, the user is warned about the automatic removal of any underwriting formulas attached previously.

User Journeys

FTOS_IP_InsuranceProductCoverFormulaUnd

General description: This default form driven flow is based on the **FTOS_IP_InsuranceProductCoverFormulaUnd** entity. The flow is used for inserting or updating underwriting formulas attached on insurance coverage level. The fields **Insurance Product Item**, **Context Type** and **Formula** are mandatory to be completed. When clicking the **Map Data** button, another screen appears and the user can select an entity whose attributes can be used as inputs for the calculation of the appended formula.

View Default This view lists all the underwriting formulas attached on the coverage level. The user can either select an existing record - for editing, insert a new record or export the entire list.

FTOS_IP_InsuranceProductItemFormulaUnd

General description: This default form driven flow is based on the **FTOS_IP_InsuranceProductItemFormulaUnd** entity. The flow is used for inserting or updating underwriting formulas attached on the product level. The fields **Context Type** and **Formula** are mandatory to be completed. When clicking the **Map Data** button, another screen appears and the user can select an

entity whose attributes can be used as inputs for the calculation of the appended formula.

View Default This view lists all the underwriting formulas attached on the product level. The user can either select an existing record - for editing, insert a new record or export the entire list.

On Demand Scripts

FTOS_IP_InsuranceProductCoverFormulaUnd_ CreateDataMapping

This script inserts a new record in the **FTOS_CALC_DataMapping** with the data received in the underwriting formula - for the coverage. More than that, the script also updates the formula with the Id of the newly inserted data mapping calculation.

FTOS_IP_InsuranceProductItemFormulaUnd_ CreateDataMapping

This script inserts a new record in the **FTOS_CALC_DataMapping** with the data received in the underwriting formula - for the product. More than that, the script also updates the formula with the Id of the newly inserted data mapping calculation.

Endpoints

FTOS_IP_InsuranceProductCoverFormulaUnd_ CreateDataMapping

It calls the **FTOS_IP_InsuranceProductCoverFormulaUnd_
CreateDataMapping** on demand script. (See above.)

FTOS_IP_InsuranceProductItemFormulaUnd_ CreateDataMapping

It calls the **FTOS_IP_InsuranceProductItemFormulaUnd_
CreateDataMapping** on demand script. (See above.)

Server Side Library

FTOS_IP_InsuranceProduct_Operations

This server side script library stores methods for various actions that a user can perform on an insurance product, or coverage. These methods were divided in two main classes, in order to keep the functionalities on the product separate from the ones on the product items (coverages).

Class InsuranceProduct

This class holds methods only used for operations performed at the product level.

getProductIdOnProductItem

This function returns a list with only one product Id that was found based on the existing lookup on a specific insurance product item (coverage) Id.

Input parameters: `productItemId` - The Id of the insurance product item (coverage).

Output parameters: List with the product Id.

getProductUnderwritingType

This function returns a list with only one underwriting type that was found for a product with a specific Id.

Input parameters: `productId` - The Id of the insurance product.

Output parameters: List with the product underwriting type.

validateProductUwType

This function returns an object that will contain the underwriting type set on a specific insurance product

Input parameters: `productId` - The Id of the insurance product for which the underwriting type will be returned.

Output parameters: `validationObj` - The object containing the underwriting type.

getProductTariffType

This function returns a list with only one tariff type that was found for a product with a specific Id.

Input parameters: `productId` - The Id of the insurance product.

Output parameters: List with the product tariff type.

validateProductTariffType

This function returns an object that contains the tariff type set on a specific insurance product.

Input parameters: `productId` - The Id of the insurance product for which the tariff type is returned.

Output parameters: `validationObj` - The object containing the tariff type.

deleteProductItemFormulaByProductId

This function deletes the formulas attached on a specific insurance product.

Input parameters:

- `entityNamesList` - The list of entities.
- `insuranceProductId` - The Id of the insurance product.

Output parameters: N/A.

Class InsuranceProductItem

This class holds methods only used for operations performed at the coverage level.

getItemFormulaOnContext

This function returns a list with underwriting formulas attached on coverage level - for a specific product Id, insurance item Id and context type Id.

Input parameters:

- `insuranceProductId` - The Id of the specific insurance product item (coverage).
- `insuranceProductId` - The Id of the specific insurance product.
- `contextTypeId` - The Id of the specific context type.

Output parameters: List with underwriting formulas.

formulaBeforeActionValidations

This function validates if the newly added formula contains a unique combination of insurance product, insurance product item (coverage) and context type. If the data is duplicated or the formula name already exists, this function throws an error.

Input parameters:

- `coverageFormulaName` - The name of the formula which is about to be inserted or updated.
- `insuranceProductId` - The Id of the selected insurance product.
- `insuranceProductId` - The Id of the selected insurance product item (coverage).
- `contextTypeId` - The Id of the selected context type.

- **maxNumber** - The maximum number of records necessary for validation.

Output parameters: N/A.

ifFormulaHasChangedUpdateDataMapping

This function checks the attached underwriting formula for updates. Next, if it finds any change, the function updates the specific record from the **FTOS_CALC_DataMapping** entity.

Input parameters:

- **formulaId** - The Id of the selected formula.
- **insuranceProductItemId** - The Id of the selected insurance product item (coverage).
- **contextTypeId** - The Id of the selected context type.
- **dataMappingId** - The Id of the related data mapping record.

Output parameters: N/A.

createDataMapping

This function is called when the user clicks the **Map Data** button for inserting a new record in the **FTOS_CALC_DataMapping** entity, with some data from the underwriting formula. It also updates the related formula with the new data mapping Id.

Input parameters:

- **formulaId** - The Id of the selected formula.
- **contextTypeId** - The Id of the selected context type.

Output parameters: N/A.

insertNewDataMapping

This function inserts a new record in the **FTOS_CALC_DataMapping** entity, with some data from the underwriting formula.

Input parameters:

- **ipFormulaData** - The data object for the selected formula.
- **formulaId** - The Id of the selected formula.

Output parameters: N/A.

uwFormulaVersioningHelper

This function runs some validations before calling the **insertNewDataMapping()** function.

Input parameters:

- **insuranceProductData** (ipFormulaData) - The data object (containing details of the insurance product) for the selected formula.
- **insuranceProductItemId** - The Id of the selected insurance product item (coverage).
- **contextTypeId** - The Id of the selected context type.
- **formulaId** - The Id of the selected formula.

Output parameters: N/A.

getInsuranceProductItemDetails

This function returns a list with some data objects for the insurance coverages found.

Input parameters: **productId** - The Id of the selected insurance product.

Output parameters: List with insurance coverages.

getInsuranceProductItemDetails

This function returns different attributes from the **FTOS_IP_InsuranceProductItem** entity, based on the product id.

Input parameters: `productId` - The Id of the selected insurance product.

Output parameters: `query` - An array that contains an object with the following results: `FTOSIPIInsuranceProductId`, and `Name`.

getInsuranceProductItemDetailsVersioned

This function returns different attributes from a versioned record of the **FTOS_IP_InsuranceProductItem** entity, based on the versioned product id;

Input parameters: `productId` - The Id of the selected insurance product.

Output parameters: `query` - An array that contains an object with the following results: `Percentage`.

getProductDetails

This function returns different attributes from a record of the **FTOS_IP_InsuranceProduct** entity, based on the product id.

Input parameters: `productId` - The Id of the selected insurance product.

Output parameters: `query` - An array that contains an object with the following results: `attributeVersion`, `Name`, `TariffType`, `ReferencedAttributeId`, and `ClonedProductId`.

getCoverageSplitDetails

This function returns different attributes from a record of the **FTOS_IP_PremiumCoverageSplit** entity, based on the product coverage id and product formula id.

Input parameters:

- **productId** - The Id of the product coverage.
- **productFormulaId** - The Id of the product formula.

Output parameters: **query** - An array that contains an object with the following results: **Name**.

getSplitDetails

This function returns different attributes from a record of the **FTOS_IP_PremiumCoverageSplit** entity, based on the product coverage id.

Input parameters: **coverageId** - The Id of the product coverage.

Output parameters: **query** - An array that contains an object with the following results: **Name**, and **Percentage**.

getProductFormula

This function returns different attributes from a record of the **FTOS_IP_InsuranceProductFormula** entity, based on the product id.

Input parameters: **productId** - The Id of the insurance product.

Output parameters:

query - An array that contains an object with the following results: **Name**.

HINT

For more details about configuring formulas, see the [Business Formulas](#) documentation and this [Risk Scoring](#) tutorial.

Defining Premium Splits

There are cases when an insurance product has multiple product items (coverages) and the insurer needs to establish a premium amount per each coverage. When the tariff type is set to **Per Product**, this is done by using the **Premium Coverage Split** functionality. Thus, the insurer can set the split (the percentage from the total premium amount) that the tariff formula will use in order to calculate the premium amount per each coverage included in the product.

Premium Coverage Split Functionality Description

Following that the **Tariff Type** of a product is set to **Per Product**, when the user goes to the **Premium Amount** tab on an insurance product and adds a new insurance product formula, all the coverages included in the product become available for percentage split fitting, inside the premium coverages split grid - with the default percentage being 0, for each coverage. Inside the grid, the user can manually enter numerical values for each percentage (assigned to a certain coverage), so that the percentages add up to 100.

Premium Coverage Split Validations

The following validations are in place:

- The user can approve the product only after assigning the split percentages, so that they add up to 100.
- If the user adds a new coverage to the product, the premium coverage split must be updated as well, otherwise the product cannot be approved.
- The user cannot add the same coverage twice to the premium coverage split.
- If the user deletes a coverage from a product, the coverage is automatically deleted from the premium coverage split as well.

Read below about the flows, libraries, and scripts used by this functionality.

User Journey

FTOS_IP_PremiumCoverageSplit

General description: This default form driven flow is based on the **FTOS_IP_PremiumCoverageSplit** entity. The flow is used for inserting or updating the total premium amount split - namely, the percentage for the premium amount per each coverage, to be used by the formula attached to the product.

View Default This view lists all the premium amount splits configured for an insurance product formula.

Business Workflow Configuration Actions

Draft_Approved

PremiumCoverageSplitValidations Action

To get the data for the validations, this action calls from **FTOS_IP_InsuranceProduct_Operations** server side script library the following functions:

- `getProductDetails(productId)`,
- `getProductFormula(productId)`,
- `getInsuranceProductItemDetails(productId)`,
- `getSplitDetails(coverageId)`.

It throws an error if the sum of the percentages doesn't add up to 100 or if a coverage from the product is missing from the **Premium Coverage Split** list.

VersionDraft_Approved

PremiumCoverageSplitValidations Action

To get the data for the validations, this action calls from **FTOS_IP_InsuranceProduct_Operations** server side script library the following functions:

- `getProductDetails(productId)`,
- `getProductFormula(productId)`,
- `getInsuranceProductItemDetails(productId)`,
- `getSplitDetails(coverageId)`.

It throws an error if the sum of the percentages doesn't add up to 100 or if a coverage from the product is missing from the **Premium Coverage Split** list.

Server Side Library

FTOS_IP_InsuranceProduct_Operations

From this library, the following functions are used:

- `getSplitDetails`,
- `getCoverageSplitDetails`,
- `getProductDetails`,
- `getInsuranceProductItemDetailsVersioned`,
- `getInsuranceProductItemDetails`,
- `getProductFormula`.

For more details, see them described, inside the same library, on the [Attach Business Formulas](#) page.

Mapping Formulas

This functionality allows the user to create data mappings for two types of flows - either **Policy Admin** or **Quote and Bind**. For both flows, the functionality covers the mapping process between insurance formulas (for premium calculation or underwriting) and insurance products, or product items (coverages).

In short, when clicking either the **Map Policy Data** or the **Map Quote&Bind Data** button, on UI, the user selects a master entity. This way, the user gets access to use the values of the selected entity's attributes as **formula keys inputs**, to obtain a certain output.

IMPORTANT!

Mapping data is a mandatory step in order for the system to know from where to fetch the product data necessary for the [formula engine](#) to run.

User Journey

FTOS_IP_FormulaDataMapping

General description: This form driven flow is based on the **FTOS_IP_FormulaDataMapping** entity. This entity stores details such as: the data mappings that were added for a specific insurance product, or coverage, the ID of the related formula, the ID of the formula parameter mapping, and the data type for the values that have been used. The **FTOS_IP_FormulaDataMapping** flow is used for adding a data mapping to either a **Policy Admin** or a **Quote and Bind** flow.

View **DataMappingTypeOnFormulas**

This view is displayed on each **Insurance Product Factory** formula form. The view is used to see the list of all the mappings that have been added on the displayed formula, along with their type.

On Demand Scripts

FTOS_IP_FormulaDataMapping_CheckExisting

This script checks if there is a data mapping of the same type already added on an insurance formula.

Scripts For Product Formulas

FTOS_IP_InsuranceProductFormula_ CreateDataMapping

This script gets the configuration for the insurance product premium calculation formula and passes it to the `createDataMapping()` method (see below). The method runs some validations on it. Next, it inserts the mapping into the `FTOS_CALC_DataMapping` and `FTOS_IP_FormulaDataMapping` entities.

FTOS_IP_InsuranceProductItemFormulaUnd_ CreateDataMapping

This script gets the configuration for the insurance product underwriting formula and passes it to the `createDataMapping()` method (see below). The method runs some validations on it. Next, it inserts the mapping into the `FTOS_CALC_DataMapping` and `FTOS_IP_FormulaDataMapping` entities.

Scripts For Product Items Formulas

FTOS_IP_InsuranceProductItemFormula_ CreateDataMapping

This script gets the configuration for the insurance product item (coverage) premium calculation formula and passes it to the `createDataMapping()` method (see below). The method runs some validations on it. Next, it inserts the mapping into the `FTOS_CALC_DataMapping` and `FTOS_IP_FormulaDataMapping` entities.

FTOS_IP_InsuranceProductCoverFormulaUnd_ CreateDataMapping

This script gets the configuration for the insurance product item (coverage) underwriting formula and passes it to the `createDataMapping()` method (see below). The method runs some validations on it. Next, it inserts the mapping into the **FTOS_CALC_DataMapping** and **FTOS_IP_FormulaDataMapping** entities."

Endpoint

The **FTOS_IP_FormulaDataMapping_CheckExisting** endpoint calls the **FTOS_IP_FormulaDataMapping_CheckExisting** on demand script to check if there is a data mapping of the same type already added on an insurance formula.

Server Side Library

FTOS_IP_InsuranceProductFormulas

This library contains methods that perform actions on the entities storing the formulas used by the **Insurance Product Factory** solution. The methods are split in two objects, which cover the link between the data mappings and the insurance formula-related entities (shortly described on the [Insurance Formulas](#) page), as follows:

Object: InsuranceProductFormulas

This object holds methods called from any of the insurance formulas.

findExistindMappings

This function returns a list with the existing formulas, based on the parameters that have been passed to the function and the conditions applied on them.

Input parameters:

- `entityName` - The entity name of the record that called the function.

- **dataType** - The data mapping type about to be created (possible values: **policyData** or **quoteBindData**). This is indicated (by the user) in the formula form, when pressing either the **Map Policy Data** or the **Map Quote&Bind Data** button, on UI.
- **formulaId** - The Id of the insurance formula that called the function.
- **formulaName** - The name of the formula data mapping about to get inserted.

Output parameters: The list with the existing formula mappings.

createDataMapping

This function builds up two objects using dynamic data from the insurance formula form that executes the data mapping. Next, the function takes those objects and inserts the contained data in two entities - **FTOS_CALC_DataMapping** and **FTOS_IP_FormulaDataMapping**.

Input parameters:

- **formulaEntityName** - The insurance formula entity name.
- **formulaId** - The Id of the insurance formula.
- **dataType** - The data type for the formula mapping.
- **contextTypeId** - The Id of the **Context Type**, included in the formula configuration.
- **productItemId** - The Id of the insurance product item (coverage) for which the selected insurance formula will run, if the **Tarif Type** is set to **Per Coverage**.

Output parameters: If successfully run, the method redirects the user to the newly inserted formula data mapping.

Object: CALC_DataMapping

This object holds methods performed only on the formula data mappings existing in the system.

findExistindMappings:

This function returns a list with an existing formula data mapping that matches a given Id.

Input parameters: `dataMappingId` - The Id of the data mapping.

Output parameters: A list containing the formula data mapping, found in the system.

Testing Formulas

This functionality allows the user to test premium calculation and underwriting formulas. The user must select an existing product, go to the **Test Calculations** tab, press the **Insert** button available on the grid, and add a test scenario. Depending on the scenario, the user also chooses which type of formulas are called for testing - either tariff or underwriting formulas, and picks the formula from a list. The test results are displayed right away and the user has the option to log them into the test grid.

HINT

Test calculations are performed according to the **Tariff Type** and **Underwriting Type**, set for the specified per product, or product item (coverage).

User Journey

FTOS_IP_TestScenario

General description: This form driven flow is based on the **FTOS_IP_TestScenario** entity. The form is used to configure a certain scenario type and see the output of the selected formula. The flow covers the following two situations:

- When the test **scenario type** is set on **Premium Amount**, automatically the system triggers the tariff type and the corresponding field is shown. If the tariff type is `perProduct`, the form displays the **Insurance Product Formula** field. If the tariff type is `perCoverage`, the form displays the **Insurance Product Item Formula** field.
- When the test **scenario type** is set on **Verify Underwriting**, the form displays the underwriting context type dropdown. Once a value for the **context type** is chosen, the system automatically triggers the underwriting type and the corresponding field is shown. If the underwriting type is `perProduct`, the form displays the **Insurance Product Formula UW** field. If the underwriting type is `perCoverage`, the form displays the **Insurance Product Item** field.

After completing the values from the formula and saving it, the **Calculate** button is displayed. On click, the output of the formula is displayed in a grid.

View Default This view is used to see the list of all the test scenarios that were performed for an insurance product. From this view, the user can:

- add a new test scenario by pressing the **Insert** button.
- see the details of each test scenario by double-clicking the test record.
- export the list of test scenarios - by clicking the **Export** button.
- delete a record - by clicking the **Delete** button.

Default Form Functions

The default form driven flow uses the following functions:

getInsuranceProductType

This function gets the product configuration - namely, the **tariff type** and **underwriting type** (using `ebs.callActionByNameAsync` method from **FintechOS** Client Side SDK) calling **FTOS_IP_GetInsuranceProductType** endpoint asynchronous.

Input parameters: N/A.

Output parameters: N/A.

getTestScenarioOutputs

This function changes some form fields to read-only if `formData.mode` is in **edit mode** and formula output was calculated and saved in **FTOS_IP_TestScenarioOutputid**.

Input parameters: N/A.

Output parameters: N/A.

getInsuranceProductDetails

If `formData.mode` is in **insert mode**, this function produces the name of the test scenario by concatenating the insurance product name and the current date. After that, it calls on the `TestScenarioTypeChange` function with a test scenario type value.

Input parameters: N/A

Output parameters: N/A

onTestScenarioTypeChange

Based on scenario type, underwriting type and product type, a set of rules and validations are set to show or hide some of the fields and views according to the requirements of the test scenario. (See the description in the user journey, above.)

Input parameters: `data` - The object provided by `ebs.addFormChangeEvent` method

Output parameters: N/A.

onInsuranceFormulaChange

This function is triggered after changing a value (using `ebs.addFormChangeEvent` method from **FintechOS** Client Side SDK) either for the **Insurance Product Item Formula** field, or for the **Insurance Product**

Formula field. Once triggered, the function calls the `getCalculationDetails` function with the currently selected value. Also, a set of rules and validations is set in this function, in order to show or hide some of the fields and views.

Input parameters: `data` - The object provided by `ebs.addFormChangeEvent` method

Output parameters: N/A.

onInsuranceFormulaUndChange

This function is triggered after changing a value (using `ebs.addFormChangeEvent` method from **FintechOS** Client Side SDK), either for the **Insurance Product Item Underwriting** field, or for the **Insurance Product Underwriting** field. Once triggered, the function calls the `getCalculationDetails` function with the currently selected value. Also, a set of rules and validations is set in this function, in order to show or hide some of the fields and views.

Input parameters: `data` - The object provided by `ebs.addFormChangeEvent` method

Output parameters: N/A.

oninsuranceProductItemIdShow

This function is triggered after changing the value (using `ebs.addFormChangeEvent` method from **FintechOS** Client Side SDK) from the **Underwriting Context Type** field. Also, a set of rules and validations is set in this function, in order to show or hide some of the fields and views.

Input parameters: `data` - The object provided by `ebs.addFormChangeEvent` method

Output parameters: N/A.

getCalculationDetails

Based on the selected items from the testing scenario form and from the tariff configuration from the insurance product form, this function fetches the necessary data from the formula to be tested, using `ebs.callActionByNameAsync` method from **FintechOS** Client Side SDK, calling **FTOS_IP_GetInsuranceProductCalculationDetails** endpoint asynchronous. The functions populates the **dataContainer** grid with the values returned from the formula.

Input parameters: `insuranceProductItemFormulaId` - The Id provided by `addFormChangeEvent`.

Output parameters: N/A.

setDefaultValue

Based on the formula `masterType`, `ArgumentType` and `objectProperties`, this function sets a default value type for each formula key, displayed in the formula grid.

Input parameters:

- `masterType` - The formula `masterType` display name.
- `subType` - The formula `ArgumentType` display name.
- `objProps` - The object properties.

Output parameters: depending on the formula key type, this function can return different types of data: Boolean, string, array, etc.

calculatePremiumAmount

This is an **on click** function.

The function calculates the output of the formula and displays it in a grid, based on the provided data. Using `ebs.callActionByNameAsync` method from **FintechOS** Client Side SDK, this function calls either the **FTOS_IP_CALC_TestVerifyUnderwriting**, or the **FTOS_IP_CALC_TestPremiumAmount**

endpoint asynchronous (depending on what is selected on the scenario type), with parameters provided by `constructEndPointParams` function. Next, the function calculates the output of the formula and displays it in a grid.

Input parameters: N/A.

Output parameters: N/A.

batchFileTmp

This is an **on click function()**. When clicking **Batch File Template** button, an Excel file gets exported with a name in the following format: `batchFileTemplate` + the current date and time. This file contains the formula input keys as headers and a single row with some dummy data that helps the user to fill in values in the right format and type. Besides these input keys, the generated file contains two more columns for adding an index to each row, and also an expected result for the test values.

Input Parameters: N/A.

Output Parameters: N/A.

runTmp

This is an **on click function()**. When the user clicks the **Run** button, the data inside the uploaded file gets processed by a function that builds an object with some form data and sends it to the `FTOS_IP_ProcessTestBatchFile` endpoint. If the endpoint result is positive and the data is successfully processed, the function continues by saving the form asynchronously. It also displays an info message.

Input Parameters: N/A.

Output Parameters: N/A.

downloadResults

This is an **on click function()**. The function calls the `getTestOutputs()` function when the **Download results** button gets clicked.

Input Parameters: N/A.

Output Parameters: N/A.

onTestingTypeChange

Depending on the selected value of the Testing type field, this function hides or shows parts of the form.

Input Parameters: N/A.

Output Parameters: N/A.

getGridData

Based on different combinations of scenario type and underwriting type values, this function calls the `onInsuranceFormulaUndChange()` or the `onInsuranceFormulaChange()` methods with different value objects as parameters.

Input Parameters: N/A.

Output Parameters: N/A.

exportToCsv

This function executes the following:

- Checks the value of the `filename` parameter.
- Based on the results, builds up the array that populates the file.
- Processes the obtained array into the desired format.
- Generates the required Excel file.
- Attaches the file path to a newly added anchor tag, used for the download part.

Input Parameters:

- **filename** - This parameter is used to determine the type of file that needs to be created.
- **rows** - The rows of data that will be added inside the file.

Output Parameters: N/A.

constructCsvBatchFileRes

This function gets the data rows to be added inside the results file and returns an array with a converted version of the values.

Input Parameters: **arr** - The rows of data.

Output Parameters: **csvArr** - The list with the converted version of the rows values.

constructCsvBatchFileTmp

This function gets the data rows to be added inside the template file. Next, it returns an array with two items, one for populating the headers and one for the dummy values.

Input Parameters: **arr** - The rows of data.

Output Parameters: **csvArr** - The list with all the key/ value pairs.

convertHeaderValue

This function is used for converting the format of items (on a list) into the right format - needed for generating the headers of the formula results file.

Input Parameters: **arrVal** - The list of the original format of the keys.

Output Parameters: **returnArr** - The list of the converted format of the keys.

convertValueToStr

This function goes through a list and, if it detects an object , it converts it to a string.

Input Parameters: `arrVal` - The list of values.

Output Parameters: `returnArr` - The list with the converted objects.

isNullOrEmpty

This function determines if a given value is null or empty and returns a boolean result.

Input Parameters: `value` - The value that needs to be checked.

Output Parameters: true/false

getTestOutputs

The job of this function is to call the **FTOS_IP_GetTestBatchFile** endpoint in order to obtain the list with the existing test scenario outputs for a test scenario, if any. Also, depending on the value of the parameter the function gets called with, a test results file will be generated.

Input Parameters: `downloadBatchFilesResults` - boolean

Output Parameters: N/A.

constructEndPointParams

Based on the `scenarioType`, `underwritingType` and `productType` values, this function constructs the endpoint parameters, necessary to call the async function.

Input parameters: `obj` - The `scenarioType`, `underwritingType` and `productType` values.

Output parameters: `result` - The object with the necessary parameters to call the async function.

constructEndPointParamsResult

Based on the `scenarioType`, `underwritingType` and `productType` values, this function builds and returns the endpoint parameters, necessary to call the async function.

Input parameters:

- `inputParam` - The input parameter object.
- `obj` - The `scenarioType`, `underwritingType` and `productType` values.

Output parameters: `result` - The object with necessary parameters to call the async function.

buildOutputGrid

This function prepares the display grid for the results of the formula that was previously calculated.

Input parameters: `gridData` - The array of object elements (key/ value).

Output parameters: N/A.

Endpoints

FTOS_IP_GetInsuranceProductType

Endpoint used to run the `FTOS_IP_GetInsuranceProductType` server automation script.

FTOS_IP_GetInsuranceProductCalculationDetails

Endpoint used to run the `FTOS_IP_GetInsuranceProductCalculationDetails` server automation script.

FTOS_IP_CALC_TestPremiumAmount

Endpoint used to run the **FTOS_IP_CALC_TestPremiumAmount** server automation script.

FTOS_IP_CALC_TestVerifyUnderwriting

Endpoint used to run the **FTOS_IP_CALC_TestVerifyUnderwriting** server automation script.

FTOS_IP_ProcessTestBatchFile

Endpoint used to run the **FTOS_IP_ProcessTestBatchFile** server automation script.

FTOS_IP_GetTestBatchFile

Endpoint used to run the **FTOS_IP_GetTestBatchFile** server automation script.

On Demand Scripts

FTOS_IP_GetInsuranceProductType

Based on the provided product Id, this script calls the **getInsuranceProductType** function, to fetch the insurance product type and underwriting type, and return the results.

Input parameters: **insuranceProductId** - The insurance product Id.

Output parameters: **fetchProductType** - The collection of records.

FTOS_IP GetInsuranceProductCalculationDetails

This script is called with a data object. The object calls the **getInsuranceProductCalculationDetails** function in order to return the formula (attached to the product, or coverage), based on the chosen values from test scenario form. The function includes four fetches, but only

executes the fetch specified by the values provided in the object.

Input parameters: `obj` - An object containing the following keys:

- `scenarioType`,
- `productType`,
- `underwritingType`,
- `insuranceProductItemFormulaId`,
- `insuranceProductId`,
- `insuranceProductItemId`,
- `underwritingContextTypeId`.

Output parameters: `fetchCalculationDetails` - The collection of records.

FTOS_IP_CALC_TestPremiumAmount

This script is called with a data object.

Based on the configured **tariff type**, the object calls either the `testProdPremiumAmount`, or the `testPremiumAmount` function - from the **FTOS.IP_CALC.FormulaEngineHelper** server automation script library.

Next, it receives the calculation results of the corresponding formula and saves the response in the **FTOS_IP_TestScenarioOutput** entity - if the **Save Output Data** option is selected.

Input parameters: `object` - An object containing the following keys:

- `type`,
- `inputParams`,
- `insuranceProductId`,

- `contextType`,
- `shouldSaveOutput`.

Output parameters: `result` - The formula output result.

FTOS_IP_CALC_TestVerifyUnderwriting

This script is called with a data object.

Based on the configured **context type** and **underwriting type**, this function calls either the `testUnderwriting` or the `testUnderwritingProductCover` function - from the **FTOS.IP_CALC.FormulaEngineHelper** server automation script library.

Next, it receives the calculation results of the corresponding formula and saves the response in the **FTOS_IP_TestScenarioOutput** entity - if the **Save Output Data** option is selected.

Input parameters: `object` - An object containing the following keys:

- `type`,
- `inputParams`,
- `insuranceProductId`,
- `contextType`,
- `shouldSaveOutput`.

Output parameters: `result` - The formula output result.

FTOS_IP_ProcessTestBatchFile

This on demand script gets called whenever a batch testing file has to be processed. It receives an object as a parameter containing the values of some of the form fields, necessary to determine which type of formula has to run and what the inputs are. With the result obtained from the formula test, the function goes ahead and creates a new object used for inserting a new record in the **FTOS_IP_TestScenarioOutput** entity.

Input parameters: `object` - An object containing the following: `file`, `contextType`, `insuranceProductId`, `insuranceProductItemFormulaId`, `insuranceProductFormulaId`, `insuranceProductItemId`, `scenarioType`, `underwritingType`, and `productType`.

Output parameters: N/A.

FTOS_IP_GetTestBatchFile

This script checks if there are any existing outputs for the specified test scenario. Next, it sends the query results list to the client side with the `setData()` method.

Input parameters: `object` - An object that contains one key with a boolean value.

Output parameters: N/A.

Server Side Script Library

FTOS.IP_CALC.FormulaEngineHelper

This library is used to create test scenarios and test premium calculation and underwriting formulas. From this library, the following functions are used:

testProdPremiumAmount

This function runs the formula based on the following input parameters:

Input parameters:

- `insuranceProductId` - The insurance product Id.
- `insuranceProductFormulaId` - The insurance product formula Id.
- `formulaInput` - The formula input.

Output parameters: `formulaResult.Input` - The result after running the formula.

testPremiumAmount

This function is meant to run the formula based on the following input parameters:

Input parameters:

- `insuranceProductId` - The insurance product Id.
- `insuranceProductItemFormulaId` - The insurance product item (coverage) formula Id.
- `formulaInput` - The formula input.

Output parameters: `formulaResult.Input` - The result after running the formula.

testUnderwriting

This function is meant to run the formula based on the following input parameters:

Input parameters:

- `insuranceProductId` - The insurance product Id.
- `contextType` - The context type.
- `formulaInput` - The formula input.

Output parameters: `formulaResult.Input` - the result after running the formula.

testUnderwritingProductCover

This function is meant to run the formula based on the following input parameters:

Input parameters:

- `insuranceProductId` - The insurance product item (coverage) Id.
- `contextType` - The context type.

- `formulaInput` - The formula input.

Output parameters: `formulaResult.Input` - The result after running the formula.

Demo Formulas

The **Demo Formulas** are offered by **FintechOS** so that you don't start the creation of your own insurance formulas from scratch. The following formulas are included in the **Import Formulas** optional digital asset:

Formula 1 - `Bankassurance_UW_Formula_Final`

Formula Steps

The following are the steps used for peril scoring:

Name	Exclude From Mapping	Master Type	SubType	CalculationType	Execution Order
ruleInsuredAmount	FALSE	Simple Type	Text	Normal	1
ruleSeismicZone	FALSE	Simple Type	Text	Normal	2
uwDecision	FALSE	Simple Type	Text	Normal	3

Formula Input

The following are the arguments used as input for the **Bankassurance_UW_Formula_Final** calculations: `insuredAmount` and `seismicZone`.

Data sets

The following data sets are used for calculations:

Data Set Name	Data Set Display Name	Data Set Discriminants	Data Set Values
BA_UW_SeismicZone	Bankassurance UW Seismic Zone	SeismicZone	BA_UW_SeismicZone
BA_UW_InsuredAmount	Bankassurance_UW_InsuredAmount	InsuredAmount	BA_UW_InsuredAmount
BA_UW_Surrender	Bankassurance UW Surrender	hasSurrender	BA_UW_Surrender

Formula 2 - EmbeddedBankAssurancePremiumAmountFormula

Formula Steps

The following are the steps used for premium calculation, with this formula:

Name	Exclude From Mapping	Master Type	SubType	Calculation Type	Execution Order
risksCalculatedPrices	FALSE	Collection	Whole Number	Iteration	1
baseQuotation	FALSE	Simple Type	Decimal	Normal	2
premiumAmount	FALSE	Simple Type	Decimal	Normal	3

Formula Input

The following are the arguments used as input for the formula calculations: `propertyConstructionYear`, `propertyInsuredAmount`, and `risks`.

Data sets

The following data sets are used for calculations:

Data Set Name	Data Set Display Name	Data Set Discriminants	Data Set Values
BankAssurance_RisksPrices	BankAssurance Risks Prices	ConstructionYear; RiskName	BankAssurance_RisksPrices

Formula 3 - HomeServiceBankAssurancePremiumAmountFormula

Formula Steps

The following are the steps used for premium calculation, with this formula:

Name	Exclude From Mapping	Master Type	SubType	Calculation Type	Execution Order
coefHomeAssistancePrice	FALSE	Simple Type	Decimal	Normal	1
premiumAmount	FALSE	Simple Type	Decimal	Normal	2

Formula Input

The following are the arguments used as input for the formula calculations: `buildingInsuredAmount` and `constructionYear`.

Data sets

The following data sets are used for calculations:

Data Set Name	Data Set Display Name	Data Set Discriminants	Data Set Values
BA_ HomeService_ Prices	Bankassurance Home Service Prices	ConstructionYear	BA_ HomeService_ Prices

Formula 4 - PA_Final_Premium

Formula Steps

The following are the steps used for premium calculation:

Name	Exclude From Mapping	Master Type	SubType	CalculationType	Execution Order
BaseRate	FALSE	Simple Type	Decimal	Normal	1
CoefFrequency	FALSE	Simple Type	Decimal	Normal	2
CoefAge	FALSE	Simple Type	Decimal	Normal	3
CoefAmountInsuredItem	FALSE	Simple Type	Decimal	Normal	4
ItemPremiumAmount	FALSE	Simple Type	Decimal	Normal	5

Formula Input

The following are the arguments used as input for the **PA_Final_Premium** calculations: **age**, **coverage**, **frequency** and **sumInsured**.

Data sets

The following data sets are used for calculations:

Data Set Name	Data Set Display Name	Data Set Discriminants	Data Set Values
PA_Age	PA_Age	Age	Age
PA_AmountInsuredItem	Amount Insured Item	AmountInsured; Coverage	AmountInsuredItem
PA_Base_Rate	PA_Base_Rate	Coverage	PA_Base_Rate
PA_Frequency	PA_Frequency	Frequency	PA_Frequency

Formula 5 - PropertyFormula

Formula Steps

The following are the steps used for premium calculation.

Name	Exclude From Mapping	Master Type	SubType	CalculationType	Execution Order
BaseRate	FALSE	Simple Type	Decimal	Normal	1
CoefFrequency	FALSE	Simple Type	Decimal	Normal	2
CoefAge	FALSE	Simple Type	Decimal	Normal	3
CoefAmountInsuredItem	FALSE	Simple Type	Decimal	Normal	4
ItemPremiumAmount	FALSE	Simple Type	Decimal	Normal	5

Formula Input

The following are the arguments used as input for the **PA_Final_Premium** calculations: **age**, **coverage**, **frequency** and **sumInsured**.

Data sets

The following data sets are used for calculations:

Data Set Name	Data Set Display Name	Data Set Discriminants	Data Set Values
HH_BaseRate	HH_BaseRate	Coverage	HH_BaseRate
HH_BuildingSumInsured	HH_BuildingSumInsured	Coverage; InsuredAmount	HH_BuildingSumInsured
HH_BuildingType	HH_BuildingType	BuildingType; Coverage	HH_BuildingType
HH_Construction_Year	Construction Year	Construction Year; Coverage	HH_Construction_Year
HH_Frequency	HH_Frequency	Frequency	Frequency
HH_ResistanceStructure	HH_ResistanceStructure	Coverage; Resistance Structure	HH_ResistanceStructure
HH_UsageType	Usage Type	Coverage; Usage Type	HH_UsageType

Formula 6 - PropertyFormula_UW

Formula Steps

The following are the steps used for premium calculation.

Name	Exclude From Mapping	Master Type	SubType	Calculation Type	Execution Order
BaseRate	FALSE	Simple Type	Decimal	Normal	1
CoefFrequency	FALSE	Simple Type	Decimal	Normal	2
CoefAge	FALSE	Simple Type	Decimal	Normal	3
CoefAmountInsuredItem	FALSE	Simple Type	Decimal	Normal	4
ItemPremiumAmount	FALSE	Simple Type	Decimal	Normal	5

Formula Input

The following are the arguments used as input for the **PA_Final_Premium** calculations: **age**, **coverage**, **frequency** and **sumInsured**.

Data sets

The following data sets are used for calculations:

Data Set Name	Data Set Display Name	Data Set Discriminants	Data Set Values
HH_UW_BuildingSumInsured	HH_UW_BuildingSumInsured	Coverage; InsuredAmount	HH_UW_BuildingSumInsured
HH_UW_ConstructionYear	HH_UW_ConstructionYear	ConstructionYear	HH_UW_ConstructionYear
HH_UW_ResistanceStructure	HH_UW_ResistanceStructure	ResistanceStructure	HH_UW_ResistanceStructure
HH_UW_UsageType	HH_UW_UsageType	UsageType	HH_UW_UsageType

HINT

For more details about configuring formulas, see the [Business Formulas](#) documentation and this [Risk Scoring](#) tutorial.

Digital Assets

The **Insurance Product Factory** digital solution is split into dedicated **digital assets**, to ease its maintenance. The assets group together configuration items (such as scripts, libraries etc.) that belong to a common context. In all fairness, this breaking down should make it easier to extend and update the solution. Read more about digital assets, on the [Configuration Management](#) page.

The following **Digital Assets** are included in the **Insurance Product Factory** solution:

Insurance Product Factory Data Model

SDK Code: IP_DataModel 4.5.0

Description: This asset contains the configuration items defining the data model of the **Insurance Product Factory** solution. It also contains the business workflow attached to **FTOS_INS_VersioningWorkflow** entity.

Items Contained: Actions, Attributes, BW, Config Data Definitions, Data Imports, Entities, Entity BWs, Entity Extensions, Entity Forms, Entity Views, Optionset Attributes, Workflow.

Insurance Product Factory SDK

SDK Code: IP_SDK 4.NEXT

Description: This asset contains all the **scripts** that run when creating or interacting with the products. It also includes **data import files** storing the versioning configuration, as required by the standard versioning process.

The following libraries are included:

FTOS_VersioningHelper_Edit

This client-side library handles the client-side versioning operations.

FTOS.IP_CALC.FormulaEngineHelper

This server-side library handles the integration with the formula engine, it helps with testing and with running the formulas for the insurance products. For details, check the [Test Calculations](#) and [Insurance Formulas](#) pages.

FTOS_CloneRecordVersionHelper_Insurance

This server-side library handles the versioning process for insurance products. For details about the versioning workflow, check the [Product Life Cycle](#) page.

FTOS_IP_InsuranceProductAPIs

This server-side library handles the interaction with a defined product, through API calls. For details, check the [Insurance Product Factory Endpoints](#) page.

FTOS_IP_InsuranceProduct_Clone

This server-side library handles the cloning process for insurance products. For details, check the [Cloning Insurance Products](#) page.

FTOS_IP_InsuranceProductFormulas

This server-side library handles the data mapping between insurance formulas and insurance products. For details, check the [Mapping Formulas](#) page.

FTOS_IP_InsuranceProduct_Operations

This server-side library handles the interaction with a defined product, through the user interface. For the description of this library, check the [Attaching Formulas](#) page.

Items Contained: Actions, Client Script Library, Workflows, Workflow Libraries.

IMPORTANT!

For more details about using the data import templates, see the [Importing Insurance Products](#) page.

Insurance Product Factory Form Driven Flows

SDK Code: IP_Forms 4.5.0

Description: This asset contains the **Form Driven Flows** related to the journey taken by the user when creating or adjusting an insurance product, in . The asset also contains entity views created for the entities included in the Data Model digital asset.

Items Contained: Entity Forms, Entity Views.

Insurance Product Factory Import

SDK Code: IP_Import 4.5.0

Description: This asset is optional. It contains the following **demo products**, configured for different insurance types:

- **Term Life** (TLF) - for Life and Health insurance;
- **Personal Accidents** (PA) - for Life and Health insurance;
- **Household Insurance** (HH01) - for Property insurance;
- **Embedded** (BA_EMB) - for Property insurance;
- **Pet Insurance** (PETINS) - for Pet insurance.

The asset also includes data import files to load the demo products

Items Contained: Data Import Templates, Data Config Definitions.

Insurance Product Factory Import Formulas

SDK Code: IP_Import_Formulas 4.5.0

Description: This asset is optional. It contains all the formulas (and their related data sets), needed to run scoring and pricing calculations for the demo products.

Items Contained: Formulas, Formula Inputs, Data Sets.

Insurance Product Factory Menu

SDK Code: IP_Menu 4.5.0

Description: This asset contains the **Insurance Product Factory** menu items, displayed in **Innovation Studio**.

Items Contained: Menu Items.

HINT

To inspect the **Insurance Product Factory** Digital Assets, go to **Innovation Studio** > main menu, on the left > Configuration Management > Digital Assets > **Digital Assets List** page and double-click on the desired Digital Asset to view general data about the Asset, the configuration items, related digital solutions, data import files, dependencies, and the configuration items migration section.

Glossary

Please check the **FintechOS** [Getting Started Glossary!](#)
