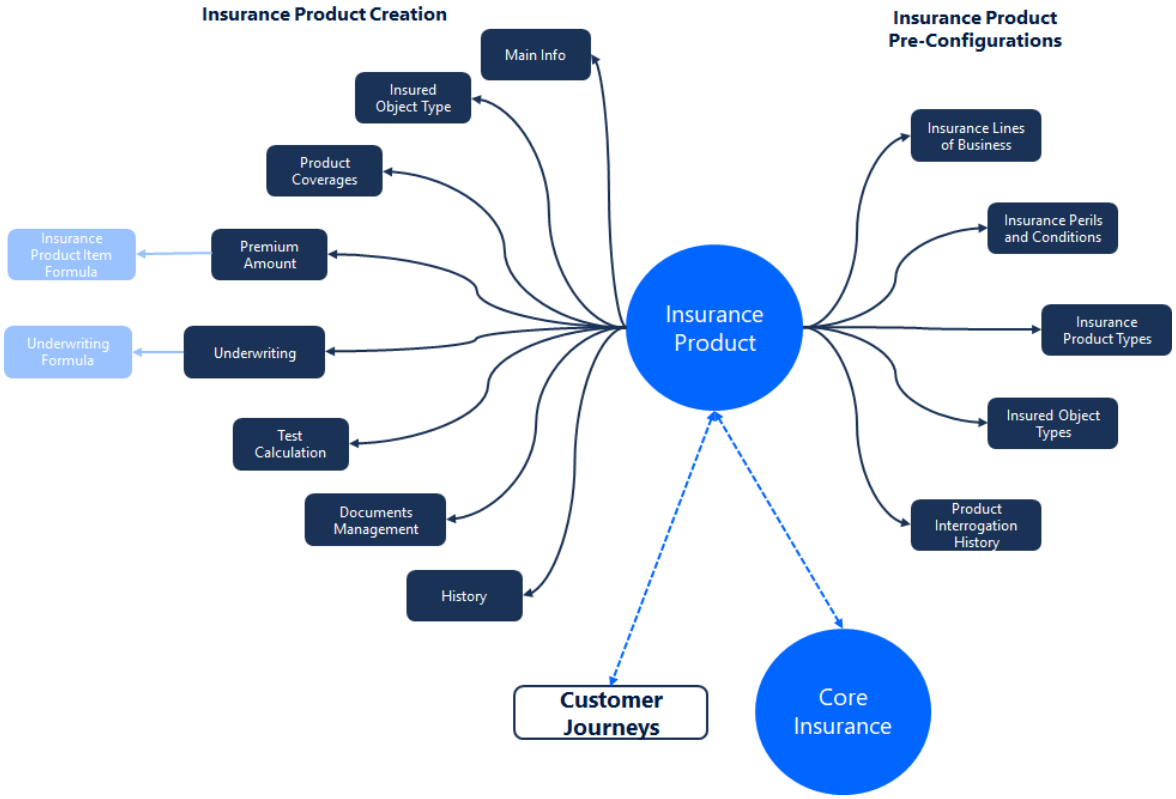
Insurance Product Factory 4.6.0

User Guide

TOC

Overview

Insurance Product Factory is an end-to-end solution that helps you grow and manage your digital portfolio. It enables you to create new insurance products, modify, or retire insurance products. The solution enables you to keep your digital portfolio accessible and comprehensible, while 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:

- Use 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.
- Use the Proposal Configurator solution on top of the **Insurance Product Factory**, in order to deliver a fully digital **Quote Configurator** customer experience. Allow the eligible customers to review different insurance products, offers, or modules and configure their own insurance quote.
- Use the Digital Journeys functionality to expose your products to your potential customers.
- Use different insurance accelerators on top of the **Insurance Product Factory** solution, in order to speed up the product delivery for specific insurance verticals.

You can also use **Insurance Product Factory** with different automation processors, that help you build your operations around the needs of your customers, and have in-depth control over the product reach - such as Omnichannel Campaigns, or Hyper-Personalization Automation, and others.

Install Insurance Product Factory

Insurance Product Factory 4.6.0 is part of the **Insurance Launch 4.6.0** package. Follow the instructions below to install and configure **Insurance Launch 4.6.0**. This is the first package to be installed in the entire insurance suite.

Prerequisites

Before installing **Insurance Launch 4.6.0**, make sure the following are already installed:

- **FintechOS Platform v22.1.4**
- **SySDigitalSolutionPackages v22.1.4001.zip**

Installation Steps

1. Download the **Insurance Launch 4.6.0** package from Release Hub.
2. Unzip your solution package .zip archive file. Identify the **01 DeploymentPackages** folder.
3. Locate the **FtosSysPkgDeployer** folder in the `FintechOS installation kit` (the path is `<unzipped_install_archive>\Tools\FtosSysPkgDeployer`). You need it to install the SySDigitalSolutionPackages.
4. Select and copy the **FtosSysPkgDeployer** folder next to the **01 DeploymentPackages** folder.

5. Create the .bat file needed for installation and save it next to the FtosSysPkgDeployer folder. Add the following script in the .bat file.

```
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
```

6. Execute the .bat file to start the installation.

.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.
- <TheNameOfTheDataBase> - The name of the database.
- <syspack_path>- The physical path to the unzipped Core Policy Admin v3.4.0 previously downloaded.
- <DB_user_password>- The password for the above mentioned SQL user.

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",
}
```

IMPORTANT!
If you decide to work with our demo products settings, download the **IPF Demo Products 4.6.0.zip** file and follow the same installation steps as for **Insurance Launch 4.6.0**.

Configure The Insurance Product Factory

The **Insurance Product Factory** makes it easy for you to create, deploy and maintain insurance products, and product sub-components. The products that you develop can transition between different business statuses, like **Draft**, **Approved**, **Closed**, or they can be updated to a new version, they can be the subject of special offers, or they can be part of recurring offers. For all these cases and more, use **Insurance Product Factory** to be on track with accurately recording said changes, control product behavior, and efficiently organize the updated information.

Lean how to define the following components to be used when creating your insurance product:

- 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;
- Business Formulas - details about how to define your insurance formulas for your products.

Configure the Lines of Business

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.

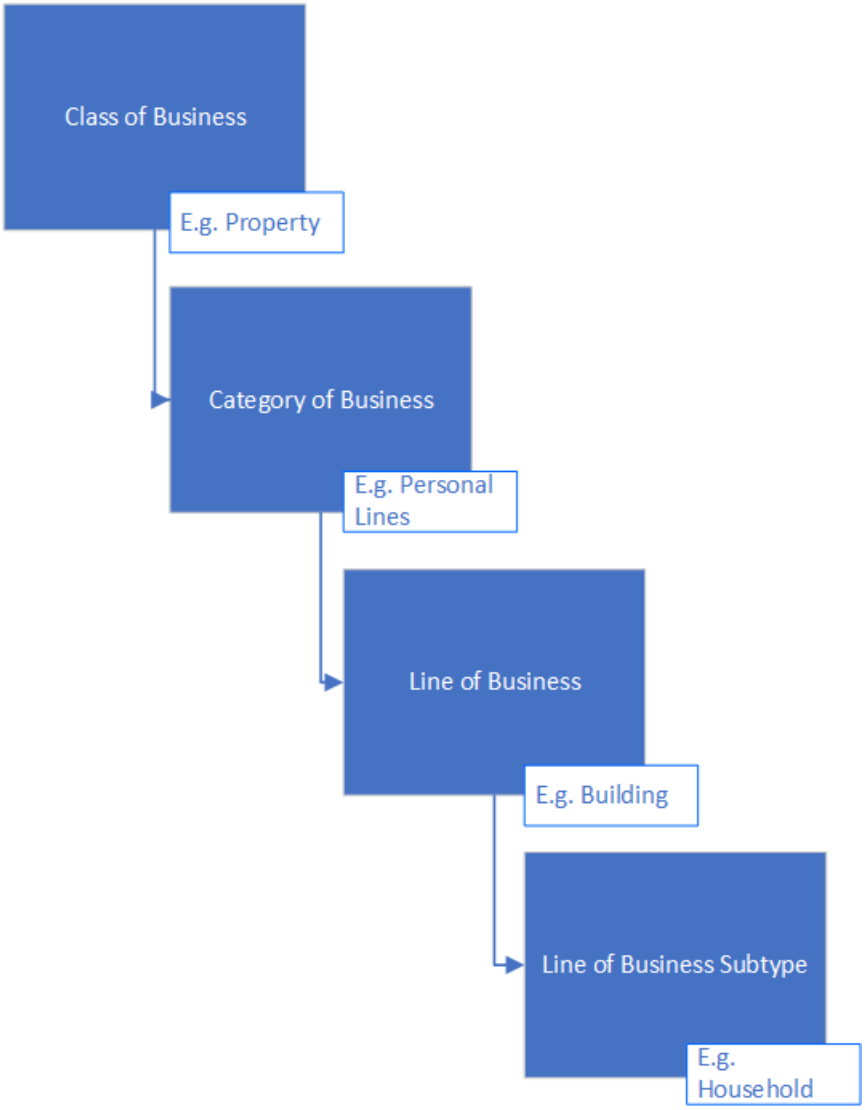
This is where the **Insurance Product Factory** comes into play: the solution has an inbuilt feature, which allows you to create and manage classification items, and hierarchies. Once defined, a classification can be attached to a product, and then the policies based on that product further inherit the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

Create Lines of Businesses

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

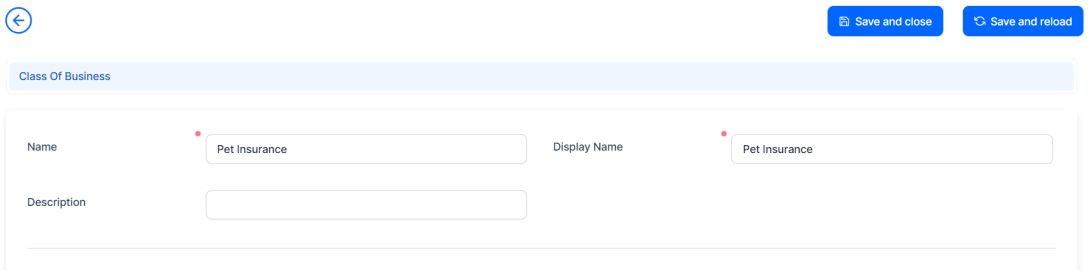
- **Class of Business** - E.g.: Property, Casualty, Life, Health;
- **Category of Business** - E.g.: Personal Lines, Commercial Lines;
- **Line of Business** - E.g.: Building, Content, Motor, Travel, Health;

- **Line of Business Subtype** - E.g.: Individual Health.

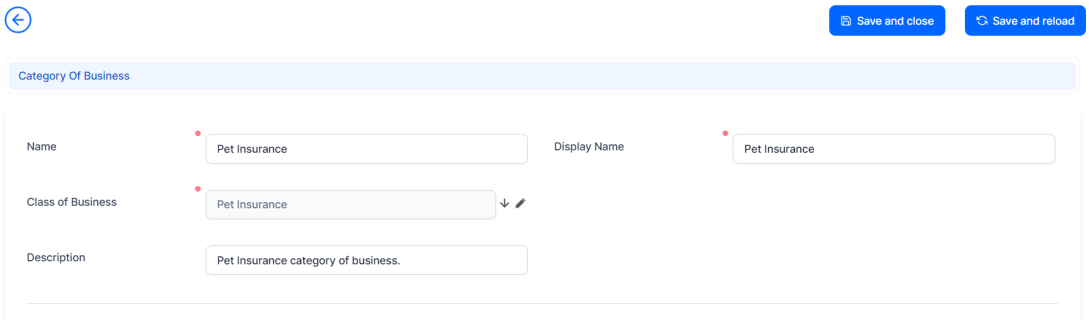


To configure your **Lines Of Business**, follow the below steps:

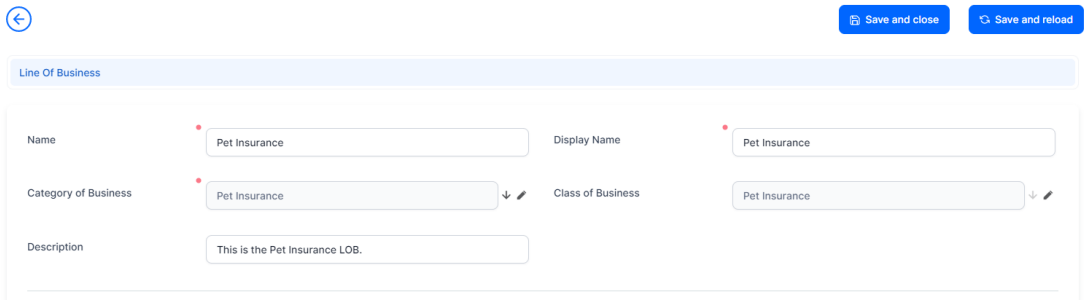
1. Navigate the main menu to **Product Factory > Product Configurator > Lines of Business**. The **Lines of Businesses** page is displayed, with the following grids: Class of Business, Category of Business, Line of Business (LOB), and Line of Business Subtype.
2. Click **Insert** in the **Class of Business** grid, to create a new record.
3. Input the details of the class of business.



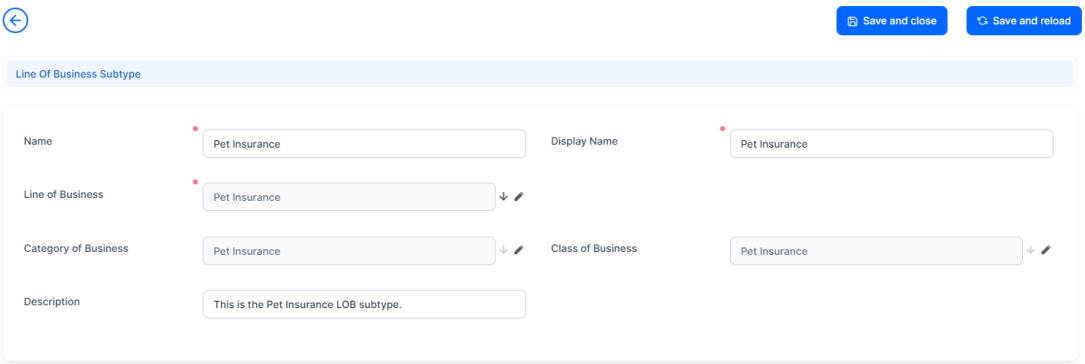
4. Click **Save and Close**. The record is displayed in the **Class of Business** grid.
5. Click **Insert** in the **Category of Business** grid, to create a new record.
6. Input the details of the category of business, and choose the Class of Business.



7. Click **Save and Close**. The record is displayed in the **Category of Business** grid.
8. Click **Insert** in the **Line of Business (LOB)** grid, to create a new record.



9. Click **Save and Close**. The record is displayed in the **Line of Business (LOB)** grid.
10. Click **Insert** in the **Line of Business Subtype** grid, to create a new record.
11. Input the details of the line of business subtype.



12. Click **Save and Close**. The record is displayed in the **Line of Business Subtype** grid.

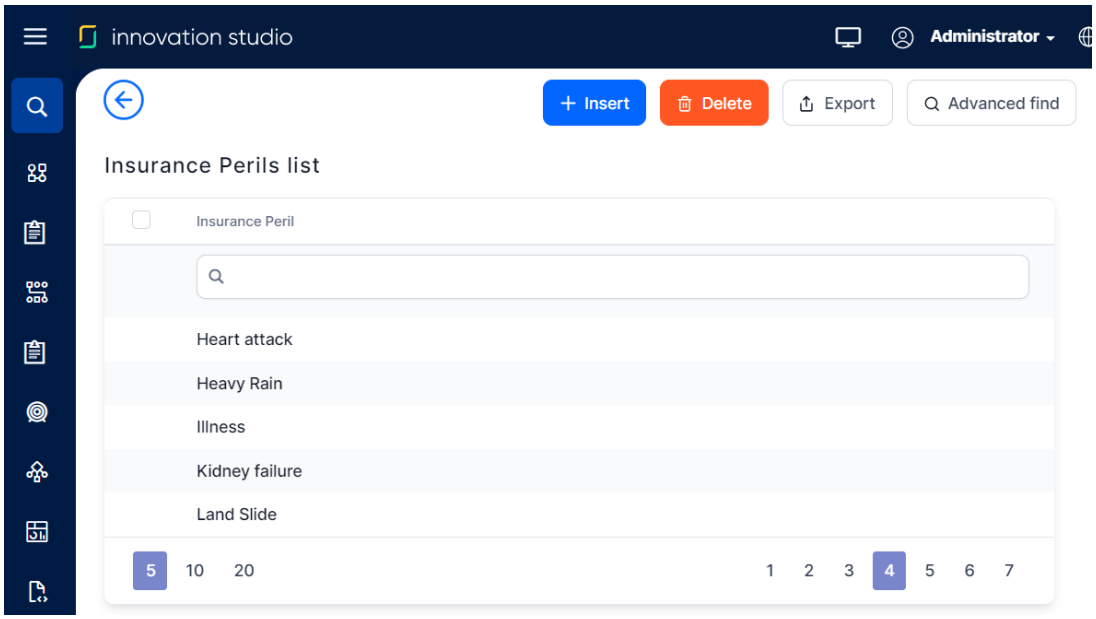
Configure the Perils

An **Insurance Peril**, or condition - for example earthquake, car accident, tornados, 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.

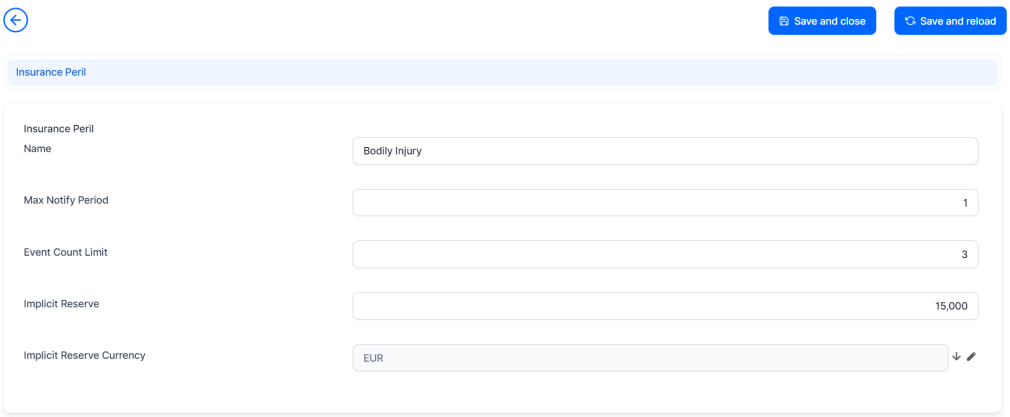
Add Perils

In **FintechOS Studio**, you can find an overview of all the insurance perils registered in your system - your nomenclature of perils, or conditions. You can also create and configure new records. Follow the steps below to add a new peril record.

1. In the main menu, navigate to **Product Factory > Product Configurator > Perils / Conditions**. The **Insurance Perils List** is displayed, containing all the existing records.



2. Click **Insert** to add a new peril.
3. Fill in the following fields to configure the peril:
    - **Insurance Peril Name**: Insert the name of the insurance peril;
    - **Max Notify Period**: Add the maximum period for the notification of the 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.



4. Click **Save and Close**. You can view the newly created record in the **Insurance Perils List**.

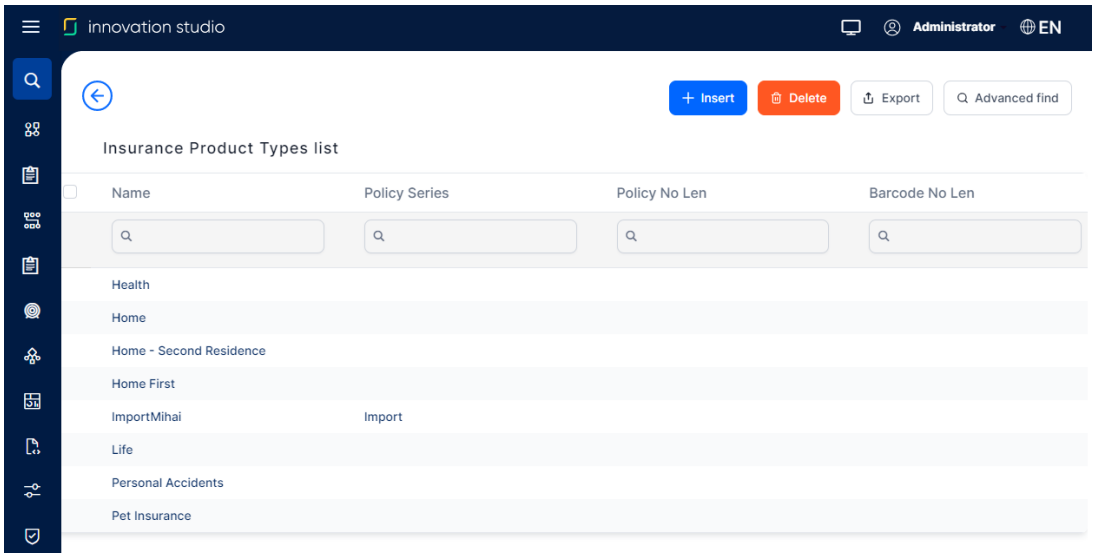
Configure the Product Types

The **Insurance Product Factory** allows you to define **Product Types** independently (e.g.health insurance, property insurance, travel insurance, or pet insurance), 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 and analysis.

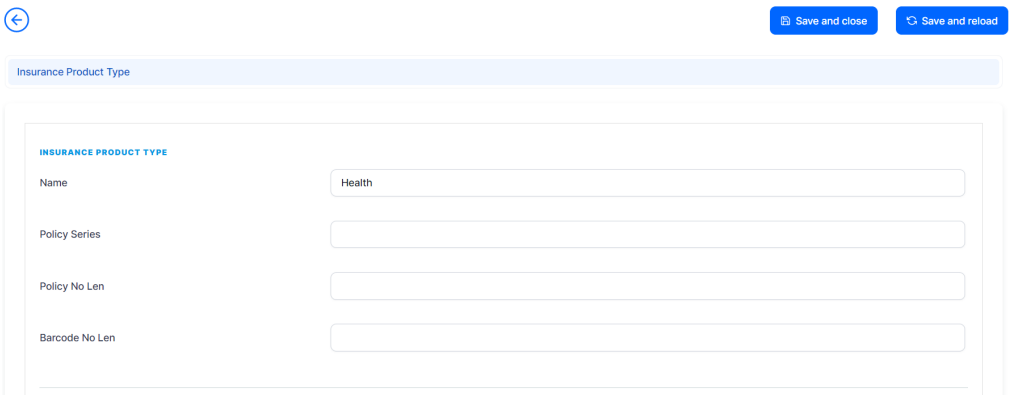
Create an Insurance Product Type

In **FintechOS Studio**, in the **Insurance Product Types** section, you have an overview of all the types registered in your system. You can also create and configure new product types. Follow the steps below to add a new insurance product type record.

1. In the main menu, navigate to **Product Factory > Product Configurator > Insurance Product Types**.
2. The insurance **Product Types List** is displayed, containing all the existing product type records.

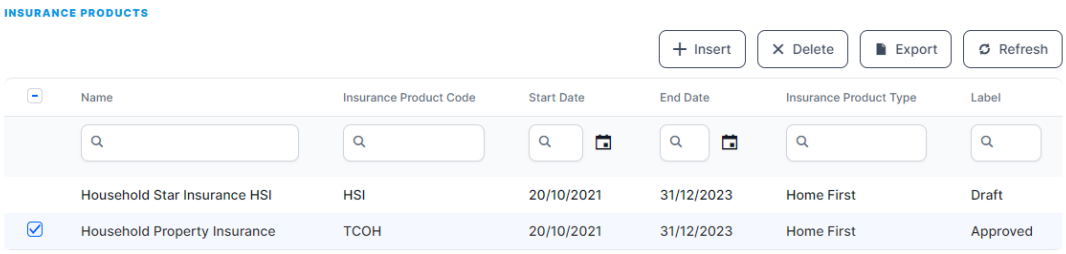


3. Click **Insert** to create a new insurance product type.
4. Fill in the following fields:
    - **Name**: Insert the name of the insurance product type;
    - **Policy Series**: Leave blank - the series of the insurance policy is presently configured through a sequencer;
    - **Policy No Len**: Leave blank - the number of digits of the insurance policy number is presently configured through a sequencer.
    - **Barcode No Len**: Leave blank.

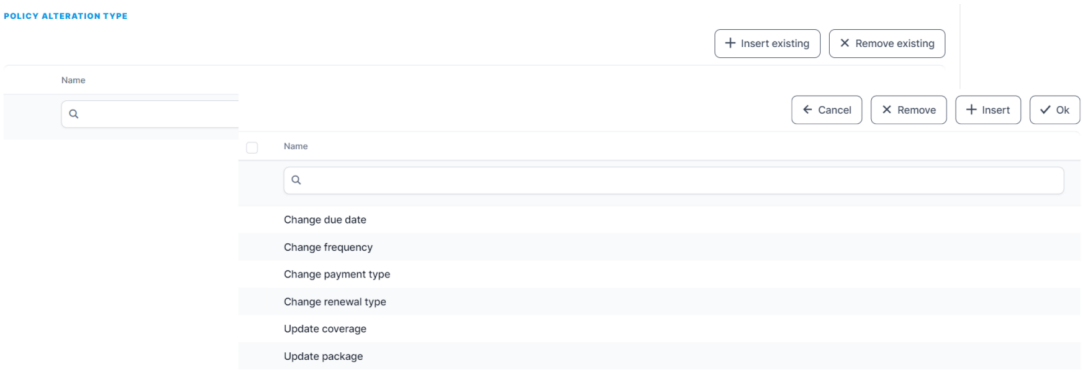


5. Click **Save and Reload**. The **Insurance Products** and Policy **Alternation Type** grids are unfolded.

6. To create an insurance product pre-filled with details from this specific product type, click **Insert** in the **Insurance Product** grid. The **Insurance Product** editor is displayed, where you need to follow the steps as per the Creating Insurance Product page. The newly created insurance product record is displayed in the **Insurance Product** grid.



7. To add alteration types for this particular product type, click **Insert** in the **Policy Alteration Type** grid.
8. Choose from the following policy alternation options: change due date, change frequency, change payment type, chane renewal type, update coverage, update package.



9. Click **OK**. The newly added options are displayed in the **Policy Alteration Type** grid.

Configure the Insured Objects

An object groups together individual properties (dimensions) that transcend an insurance product. For example, for a motorcycle, you might want to define the brand, model, and the year of manufacture, 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 status.

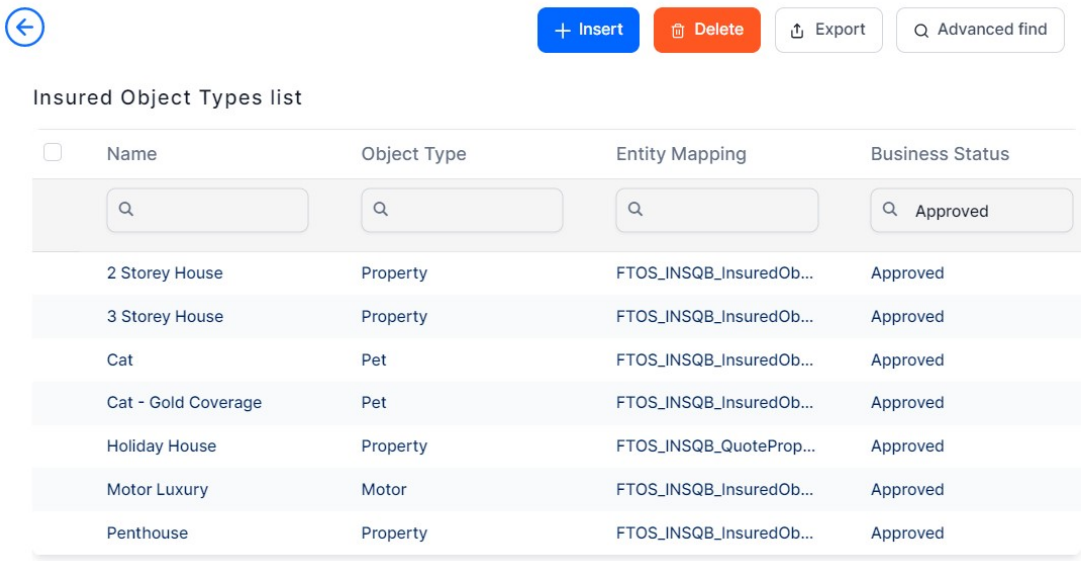
This functionality allows you to create diverse object types, from types representing very specific insurable objects - such as house objects, or car objects, to health and life objects, and even more abstract objects - such as cyber insurance objects. Additionally, your objects are independent from insurance products, so that you can easily operate with and maintain them.

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.

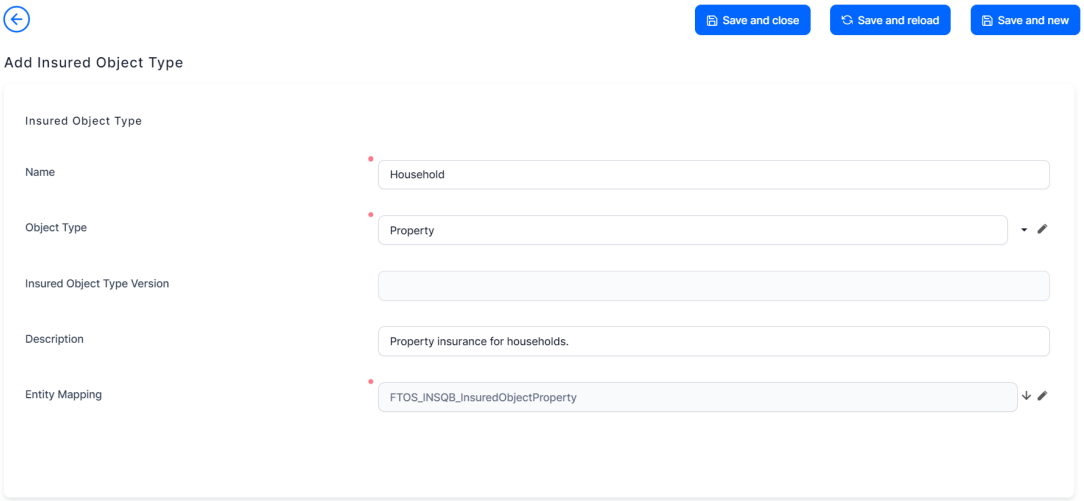
Create an Insured Object Type

In **FintechOS Studio**, in the **Insured Object Types** section, you have an overview of all the object records registered in your system. Follow the steps below to create an insured object type.

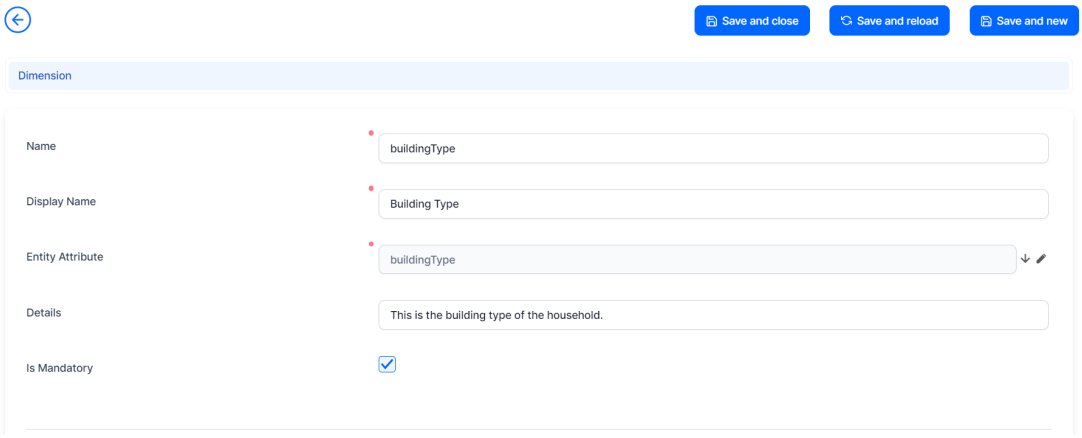
1. In the main menu, navigate to **Product Factory > Product Configurator > Insured Object Types**. The **Insured Object Types List** is displayed.



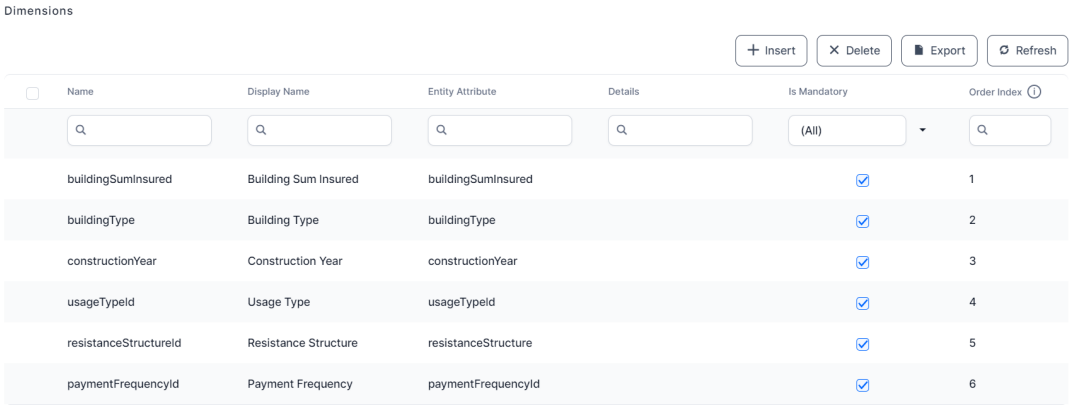
2. Click **Insert** to create a new object type. The **Add Insured Object Type** page is displayed.
3. Fill in the details for the insured object and select the right entity for it to be mapped.



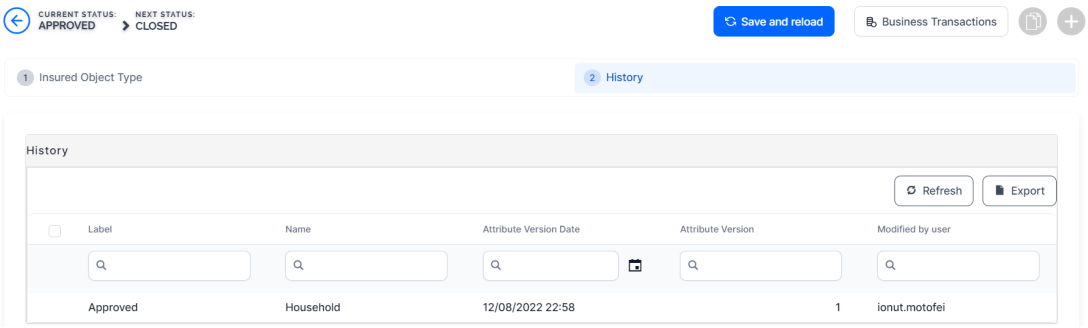
4. Click **Save and Reload**. The **Dimensions** and the **Declarations** grids are unfolded.
5. In the Dimensions grid, click **Insert** to add a new dimension for the insured object. The dimension editor is displayed.
6. Fill in the details of the insured object dimension.



7. Click **Save and Close**. All the dimensions records you create are displayed in the **Dimensions** grid.



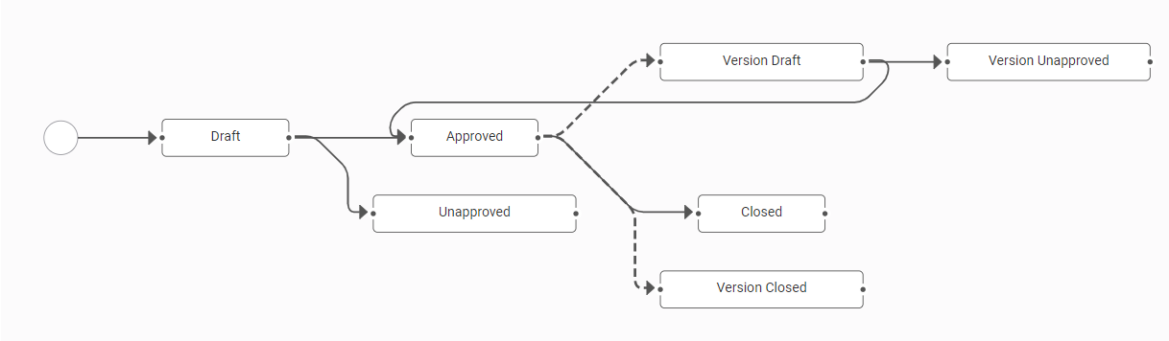
8. Access the **History** tab of the insured object to view all the versions that were created for said object.



Manage the Object Versions

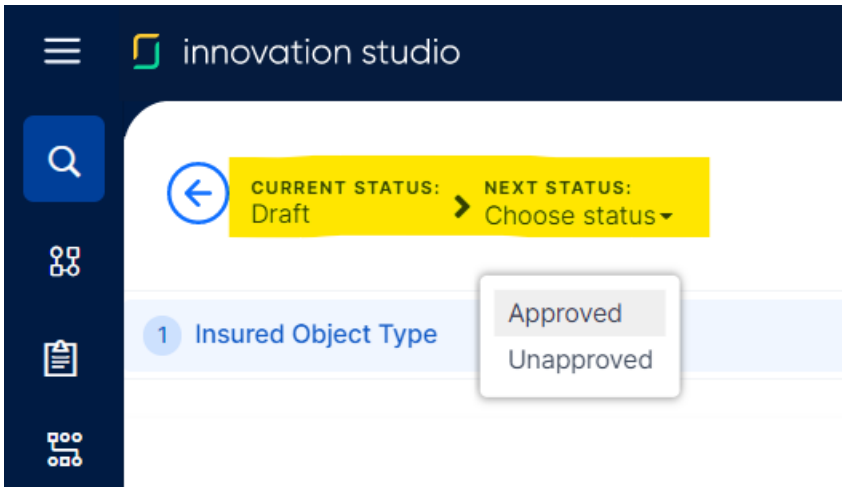
In its life cycle, the insured object type can go through the following business statuses: Draft, Approved, Unapproved, Closed, Version Draft, Version Unapproved, Version Closed.

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



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.
3. Click 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**, change the status of the newly created version from **Version Draft** to **Approved**.



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.

View the API Interrogation History

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

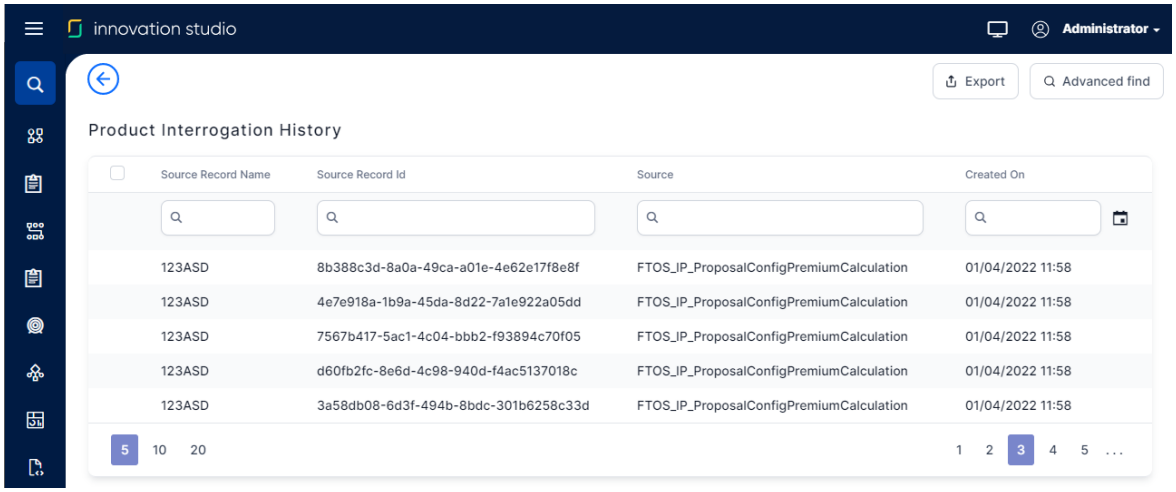
- Get Underwriting Rules Result API;
- Premium Amount API;
- Proposal Configuration Premium Calculation.

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.

The **Product Interrogation History** section is automatically updated, and you cannot add or delete records. The history of product interrogations is only available if the system parameter **Product Interogation History Enablelog** value is set to **1**.

To view the Product Interrogation History List, navigate the main menu to **Product Factory > Product Configurator > Product Interrogation History**. The existing records are displayed in a grid, as per below:



Create Insurance Products

With **Insurance Product Factory**, 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. You can configure a **product coverage** to have its own charging and underwriting structure. A product can include multiple coverages, with different configurations.

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 more details, consult 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, so you can embed it into different digital journeys and expose it to your potential customers, through different digital touchpoints, channels, or portals.

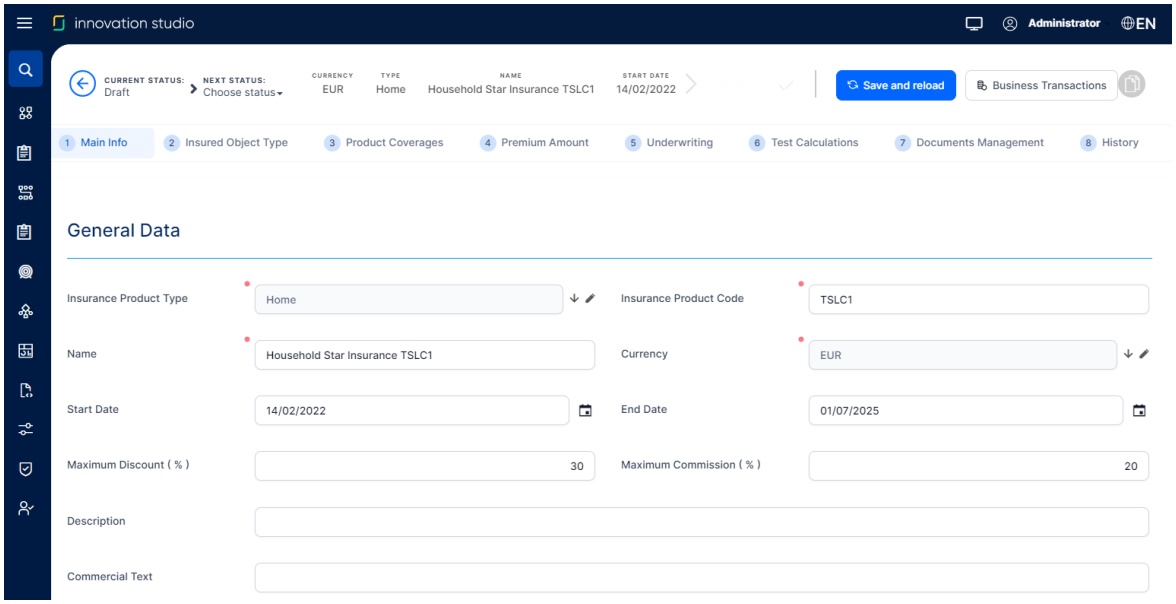
Create a new Insurance Product

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

1. In **FintechOS Studio**, navigate the main menu to **Product Factory > Insurance Products**. The **Insurance Products list** is displayed.
2. Click **Insert** to add a new record. The **Insurance Product** dynamic form opens.
3. Use the **available product configurations tabs** listed below, for creating your product:
    - Main Info - to write the general data, and the business process configuration of the insurance product;
    - Insured Object Type - to define the insurance object type and its dimensions;

- Product Coverages - to insert the main and optional coverages for the insurance product;
- Premium Amount - to insert the formula that calculates the premium amount for the coverages;
- Underwriting - to insert the formula for the underwriting of the insurance product;
- Test Calculations - to define the test scenarios for the formulas to be used for calculations;
- Documents Management - to insert any documents that need to be generated in the journey of the insurance product;
- History - to view all the versions that were created for the insurance product.

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



Define the Product's Business Details

When creating a new insurance product, 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:

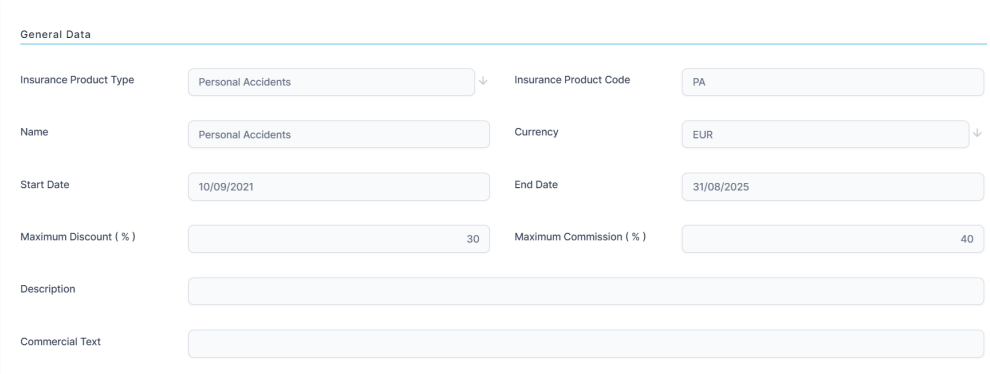
- **General Data** - This section lets you introduce the product's main characteristics, those which are most visible to the final customer.
- **Business Process Configuration** - This section lets you configure the policy coverage, policy administration, the scheduling of payments and billing, the management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

Input the General Data and Configure the Business Process

In this section, generally describe your new product.

1. Fill in the following fields:
    - **Insurance Product Type**: Select a type of insurance for your insurance product - e.g. Auto, Health, Home, Travel. See Insurance Product Types for details;
    - **Insurance Product Code**: Fill in with a code of the insurance product. If you leave the field empty and you save the record, a code is automatically generated. You cannot edit a product without having the code.
    - **Name**: Insert the name of your insurance product;
    - **Currency**: 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;

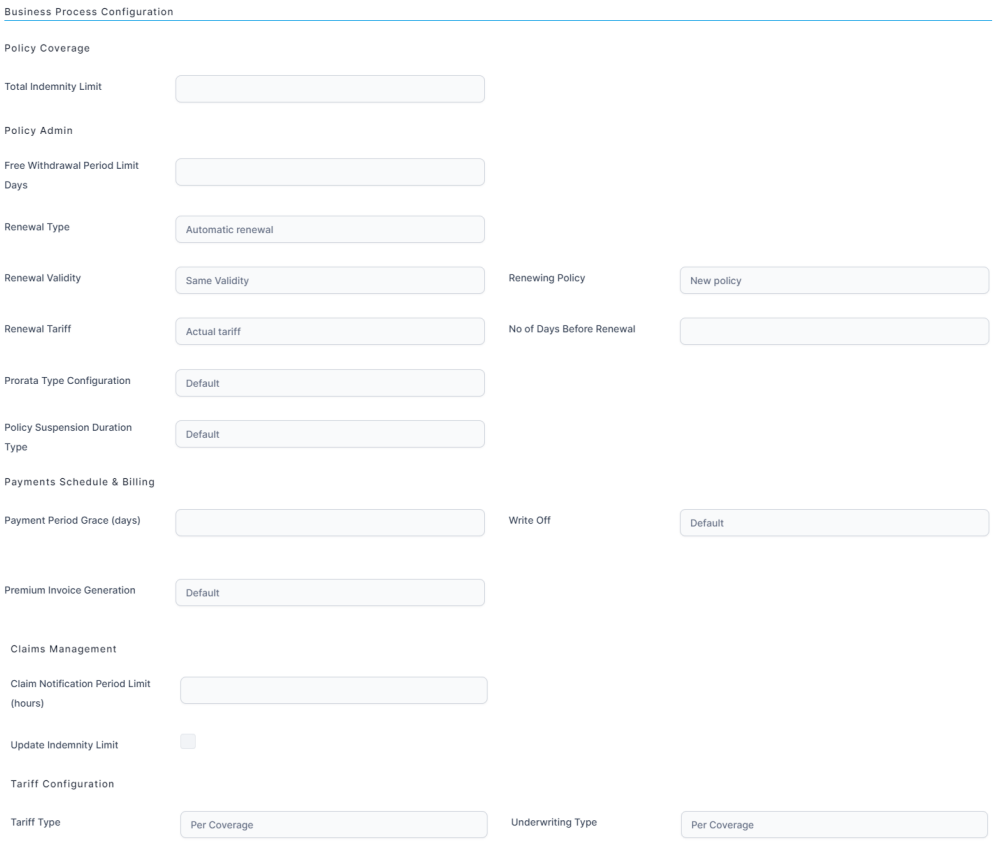
- **Maximum Discount (%)**: Set the maximum percentage of the commercial discount which can be offered in the sales process for your product;
- **Maximum Commission (%)**: Set the maximum percentage of the commission which can be offered to intermediary sellers;
- **Description**: Describe the insurance product ;
- **Commercial Text**: Input the commercial text.



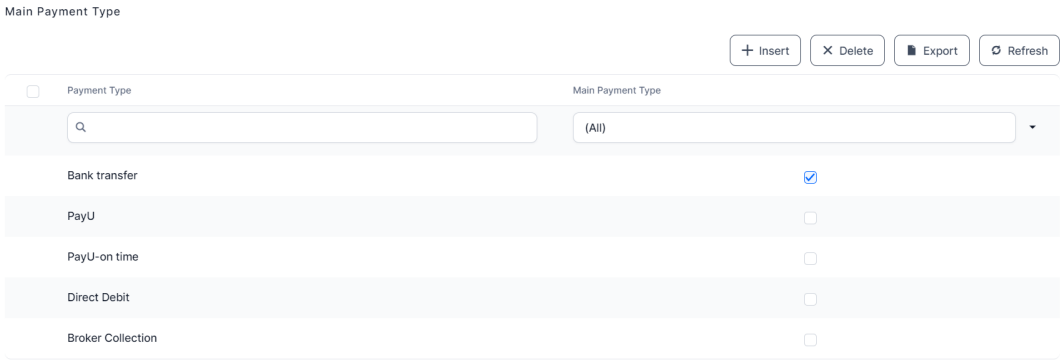
2. Move to the next section, **Business Process Configuration**, and fill in the following fields:
   - **Total Indemnity Limit**: Insert the maximum coverage amount provided per insurance policy;
   - **Free Withdrawal Period Limit Days**: Set the limit (expressed in days) for the free withdrawal period, if any;
   - **Renewal Type**: Choose the type of the insurance policy renewal.
     The option set values are: **[none]**, **No** (default), **Automatic renewal**, and **Renewal offers**.
     If you choose **Automatic renewal**, the following option set fields become available:
     - **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**: Set the proportion rate type for premium payments. The option set values are:
    - **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**.
- **Policy Suspension Duration Type**: Input the duration type for a policy suspension. It can be **Default**, as set in the general parameter, or **Specific**, where you can set up a custom value.
- **Payment Period Grace (days)**: Insert the number of days for the payment grace period;
- **Write off**: 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**.
  Choose:
    - **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 pop-up window appears and you can add your own key-value pairs, per different currencies in json file format. Set the desired values for this parameter. The values you insert are automatically saved.
- **Premium Invoice Generation**: Configure the values to be used for automatic premium invoice generation. The option set values are: **Default**, **Specific SGDAY*** and **Specific Day**.
  Choose:
    - **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.
- **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 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**.



3. Click **Save and Reload**. The**Main Payment Type** insert form is displayed.
4. Choose the **Main Payment Type** from the list, containing the following options: **Bank Transfer**, **PayU**, **PayU-on time**, **Direct Debit**, and **Broker Collection**.



5. Click **Save and Reload**. The rest of the tabs are displayed, to continue configuring the insurance product.
6. Continue to tab 2, **Insured Object Type**.

Choose the Insured Object

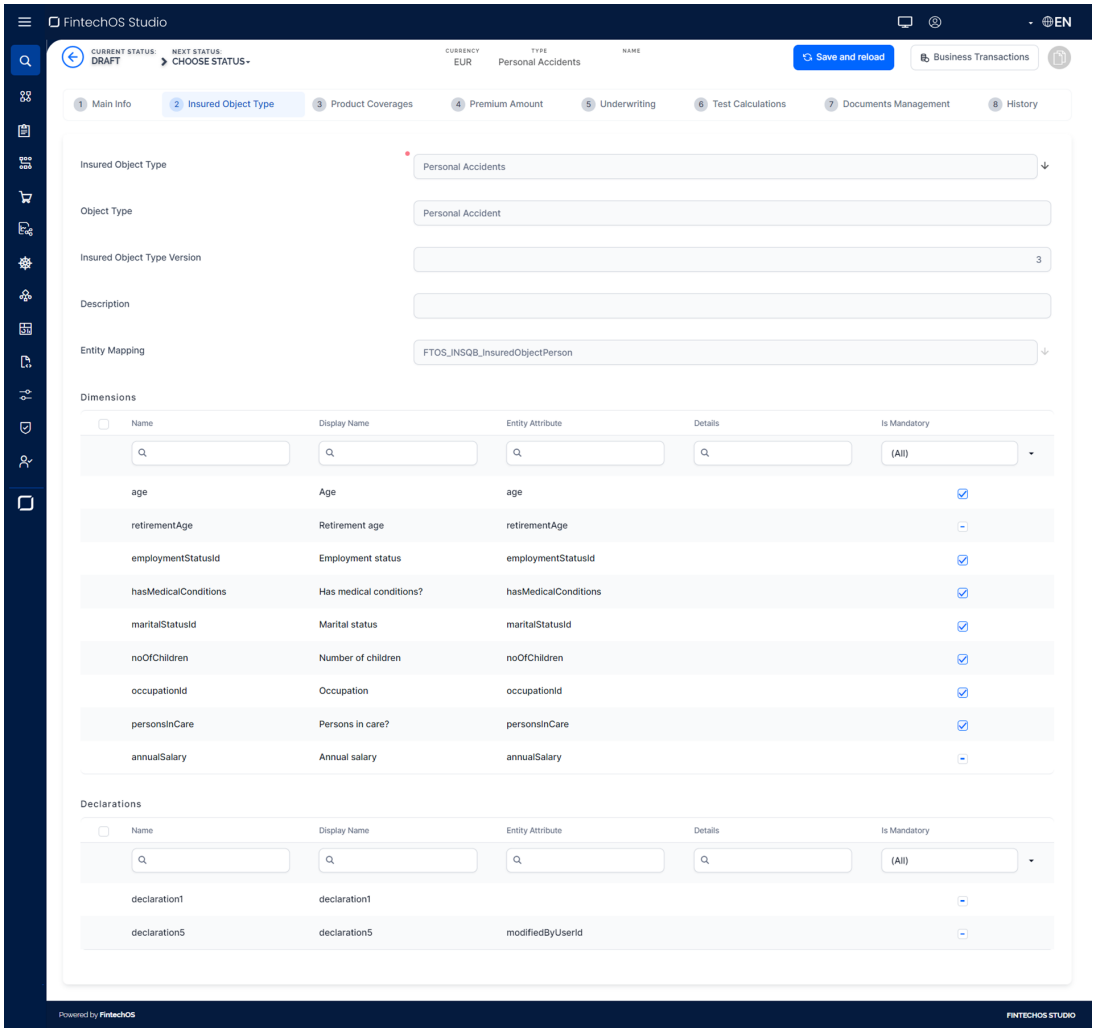
In the **Insured Object Type** you attach an object to a product, either by choosing one from your **Insured Object Types List** or by inserting a new one. After adding the object, you can see all the object dimensions, in read-only format. For more details about creating, editing, or versioning objects, see the Configure Insured Object Types page.

Add an Object to your Product

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

1. In tab 2 **Insured Object Type**, choose one of the options from the **Insured Object Type** drop-down field. The form is automatically field with the data from the already configured object, including the **Dimensions** and **Declarations**.

2. View the form. You can choose to toggle any of the dimensions or declarations to be mandatory or not for the product.



3. Continue to tab 3, **Product Coverages**.

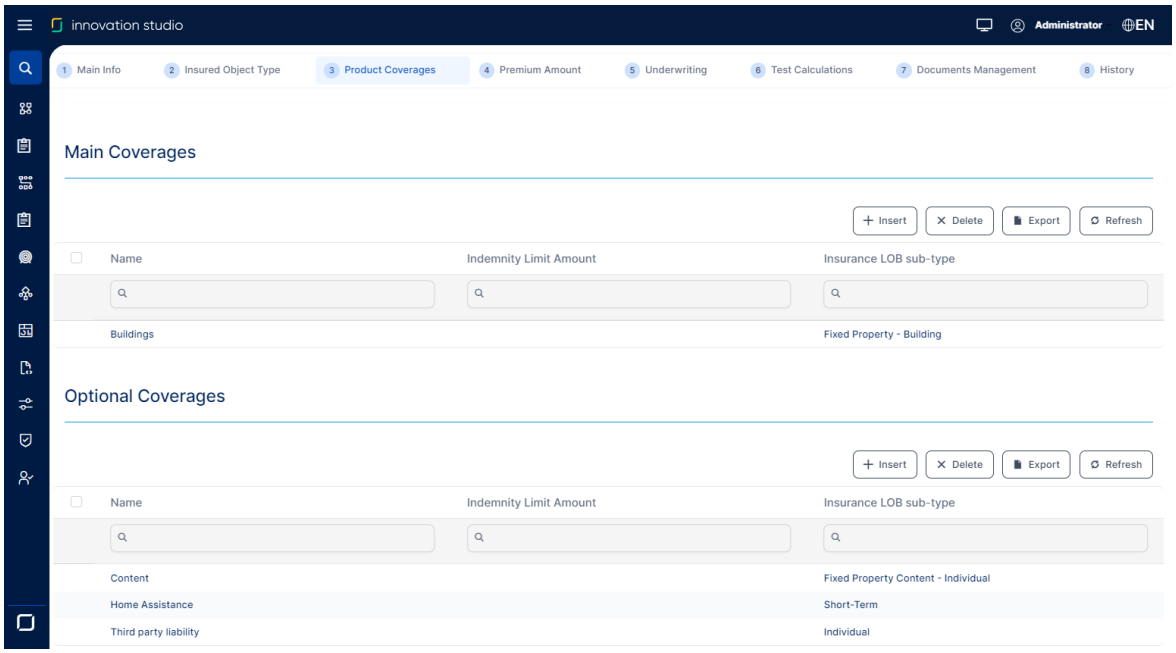
Choose the Coverages

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.

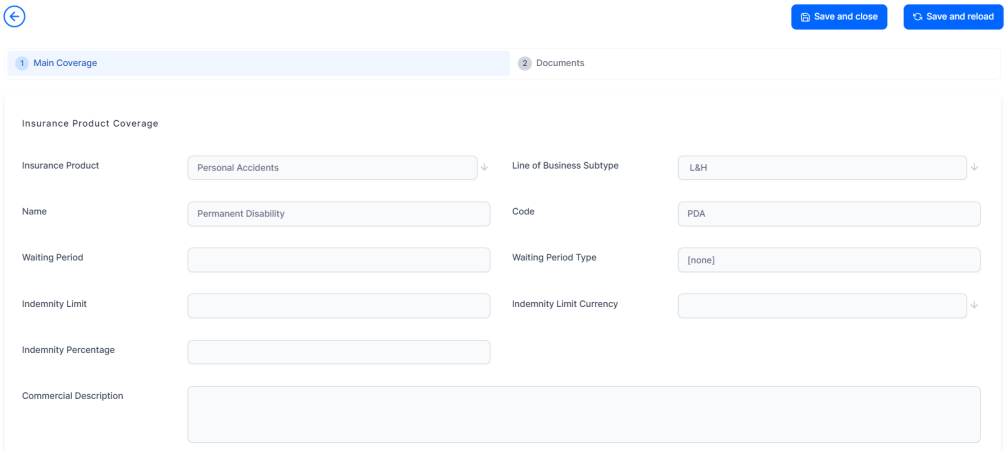
Below, an example of the available configuration sections, with some attached coverages, for a **Draft** product:



Define the Main and Optional Coverages

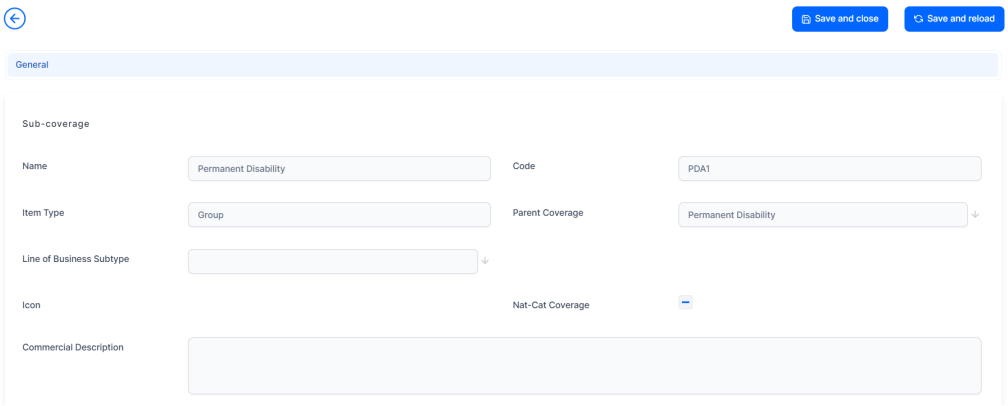
The **Main Coverage** insert form allows you to configure any number of main coverages for your insurance product. It also lets you attach all necessary documents describing each added coverage. Follow the steps below to add a main coverage:

1. In the **Main Coverages Grid**, click **Insert**. The **Insurance Product Coverage** form is displayed.
2. Fill in the following fields:
    - **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;
    - **Indemnity Limit**: The maximum monetary amount provided on the policies incorporating this main coverage;
    - **Indemnity Limit Currency**: From the drop-down list, select a currency for the indemnity limit;
    - **Indemnity Percentage**: Det the main coverage indemnity limit as a percentage from the total indemnity limit of the insurance product;
    - **Commercial Description**: Text area for describing the main coverage.

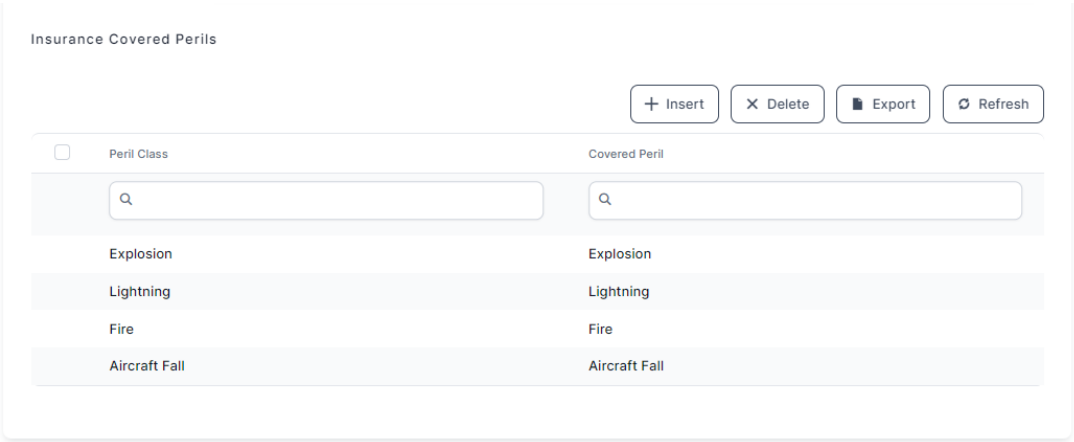


3. Click **Save and Reload**. The **Sub-coverages** grid is displayed.

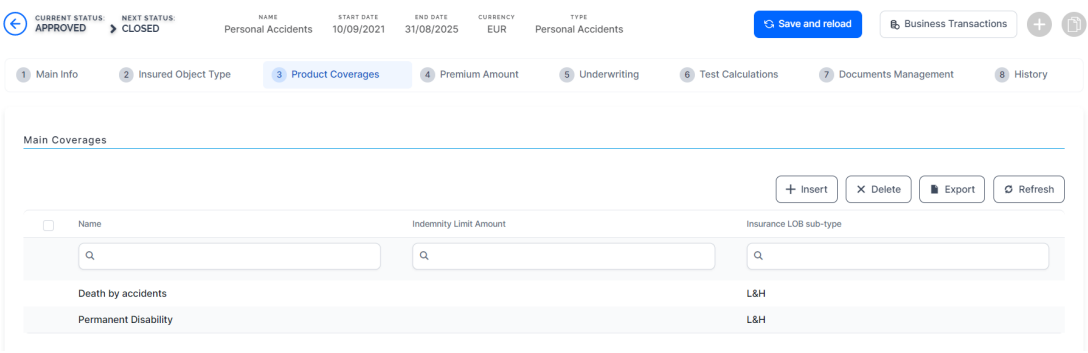
4. Click **Insert** in this grid to add a new sub-coverage. This allows you to atach perils (e.g. natural disasters), or groups of perils, to the main coverage.
5. Fill in the following fields:
    - **Name**: Add a name for the sub-coverage;
    - **Code**: Add a code for the sub-coverage;
    - **Item 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 or condition, if necessary;
    - **Nat-Cat Coverage**: Check the box if the peril or condition belongs to the Natural Catastrophes group;
    - **Commercial Description**: Text area for describing the sub-coverage.



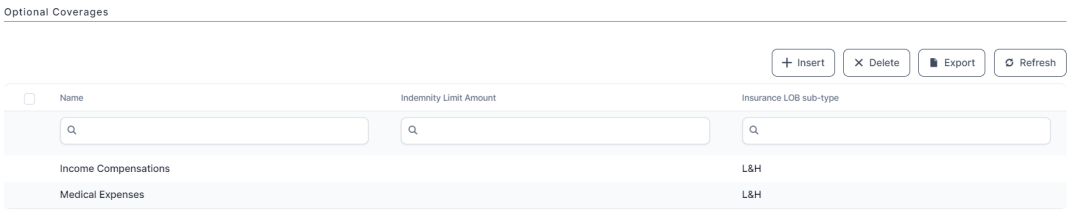
6. Click **Save and Reload**. The **Insurance Covered Perils** grid is displayed.
7. In this grid, click **Insert**. The **Insurance Covered Peril** form is displayed.
8. In the **Covered Peril** drop down, you can see a list of all the types of perils and conditions covered by that particular sub-coverage. Choose the perils you want to cover for this sub-coverage.
9. Click **Save and Close**. The chosen covered perils are displayed in the grid.



10. Click **Save and Close**. Access the **Documents** tab.
11. Click **Insert**. The **Add Document** form is displayed.
12. Fill in the fields with the name and the code of the document, and also the following details:
    - **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;
    - **Included in offer template**: Check the box if your document needs to be included in the product offer template;
    - **Included in the policy template**: Check the box if your document needs to be included in the policy template.
13. Click **Add file** to upload your document.
14. Click **Save and Close**. The file is displayed in the **Documents** grid.
15. Click **Save and Close**. The defined main coverage is displayed in the **Main Coverages** list, in the **Product Coverages** tab.



16. Move to the **Optional Coverages** grid of the **Product Coverages** tab. The **Optional Coverages** insert form allows you to configure any number of optional coverages for your insurance product.
17. In the **Optional Coverages** grid, click **Insert**. The **Insurance Product Coverage** form is displayed.
18. Follow the same steps as for the Main Coverage (2-15) to insert and configure **Optional Coverages**. The added records are displayed in the grid.



19. Continue to tab 4, **Premium Amount**.

Set the Premium Amount

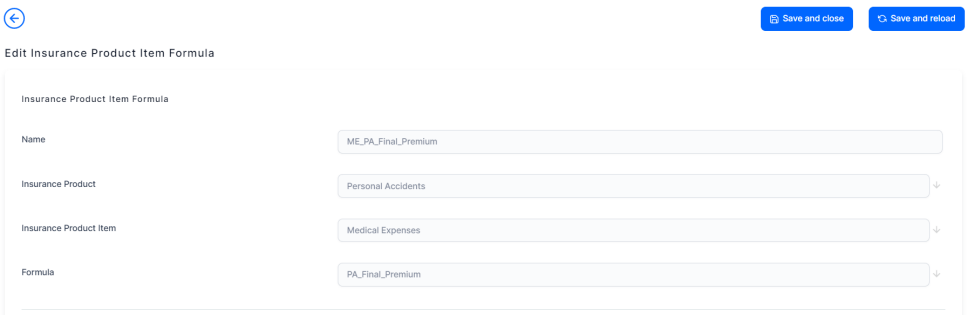
The **Premium Amount** tab allows you to configure how premium amounts are calculated for your insurance product, or product items (coverages), by attaching calculation formulas to them. You can also change previous configurations by deleting attached formulas and inserting new ones, if needed.

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

Add an Insurance Product Item Formula

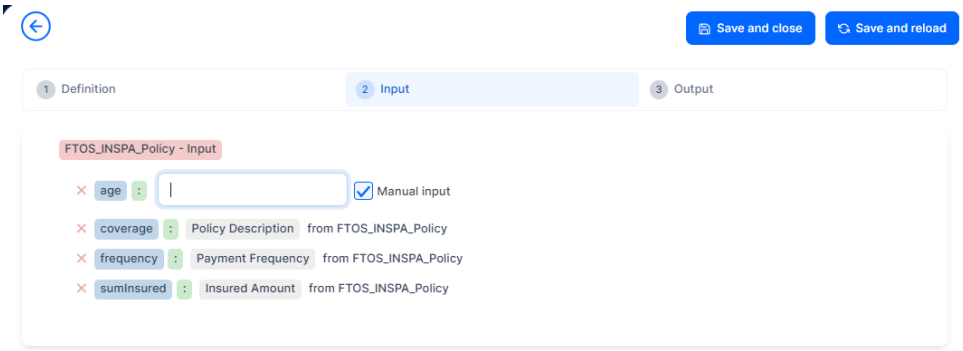
To add perform an **Insurance Premium Coverages Split**, make sure you have previously selected the **Per Product** tariff type, in the **Main Info** product tab.

1. In the **Premium Amount** tab, click **Insert**. The **Add Insurance Product Item Formula** form is displayed.
2. Insert a **Name** for the formula. The **Insurance Product** field is filled automatically.
3. In case the **Tariff Type** is set as Per Coverage, select the **Insurance Product Item** to which the formula is applied.
4. Select the **Formula** to be attached to the current product for the premium amount calculation.



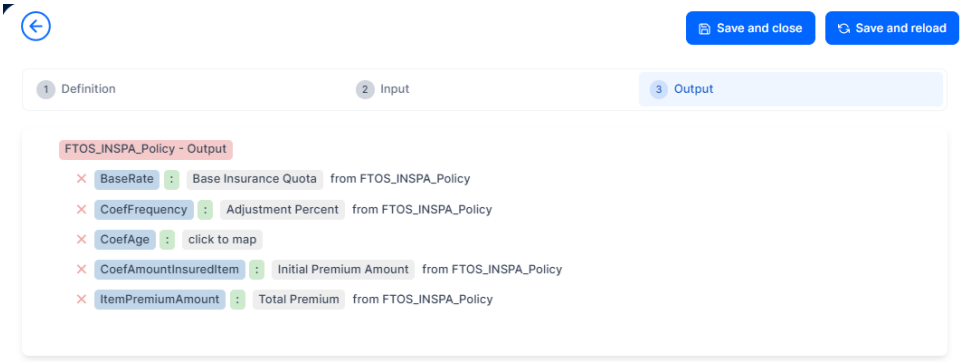
5. Click **Save and Reload**. The **Data Mapping** grid is displayed, and, if the **Tariff Type** is set as Per Product, the Insurance **Premium Coverages Split** section is displayed as well. In the **Data Mapping** grid, you can choose to define a single data mapping, for either a **Policy Admin** or a **Quote and Bind** flow.
6. Click either **Map Policy Data** or **Map Quote&Bind Data**, depending on the mapping you need to perform. The **Formula Parameter Mapping** form is displayed.
7. In the **Definition** tab, choose a **Master Entity** from which the system extracts the input keys for the formula.
8. Click **Save and Reload**.

9. Go to the **Input** 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 extracts the value needed for calculation.

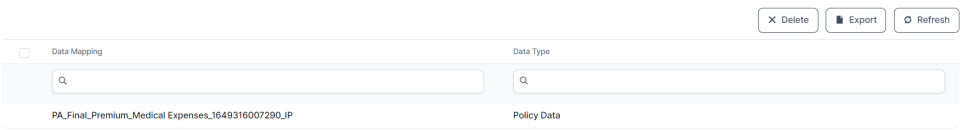


10. Click **Save and Reload**.

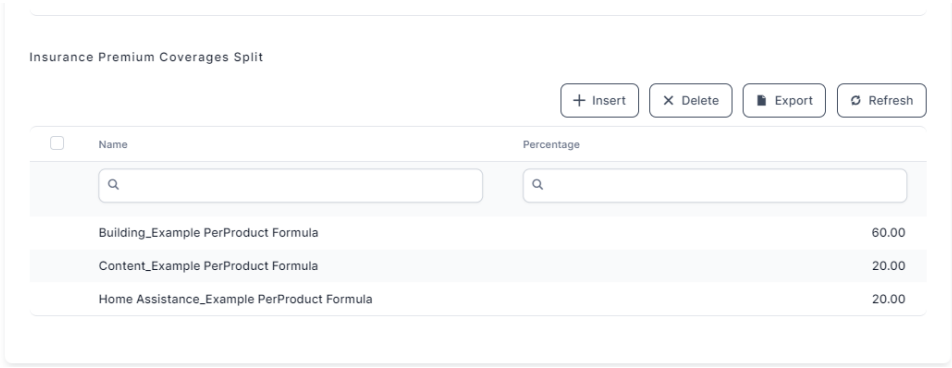
11. Go to the **Output** tab, and use the **Click to Map** buttons, next to each **formula output key** (in blue), in order to indicate the attribute to which the system makes updates (insert the value obtained) after the calculation.



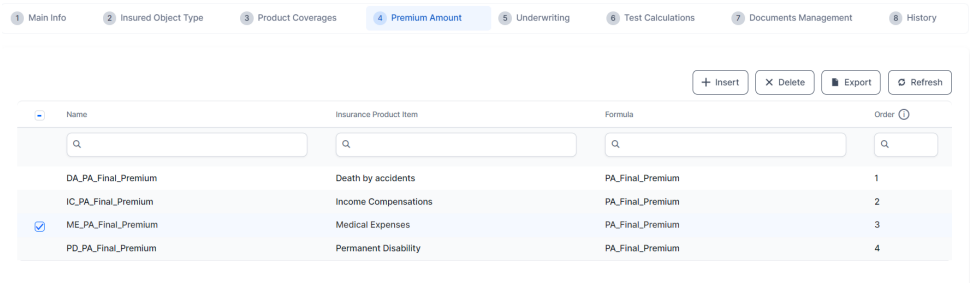
12. Click **Save and close**. You can now see the attached formula in the Data Mapping grid, at the bottom of the insert formula form, example below:



13. In the **Insurance Premium Coverages Split** grid, set the **Percentage** used to calculate the premium amount per each coverage included in the product.



14. Click **Save and Close**. You can see the configured insurance product formula in the **Premium Amount** tab.



15. Continue to tab 5, **Underwriting**.

Define the Underwriting Rules

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.

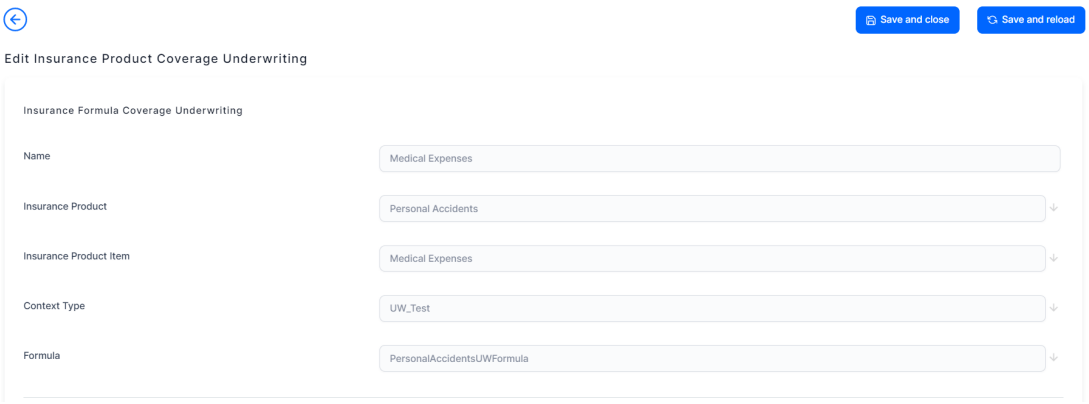
Based on the **Tariff Type** previously chosen in the **Tariff Configuration** section, you can either see the **Insurance Product Underwriting** grid or the **Insurance Product Coverage Underwriting** grid, on this tab.

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

Add an Insurance Product Underwriting Formula

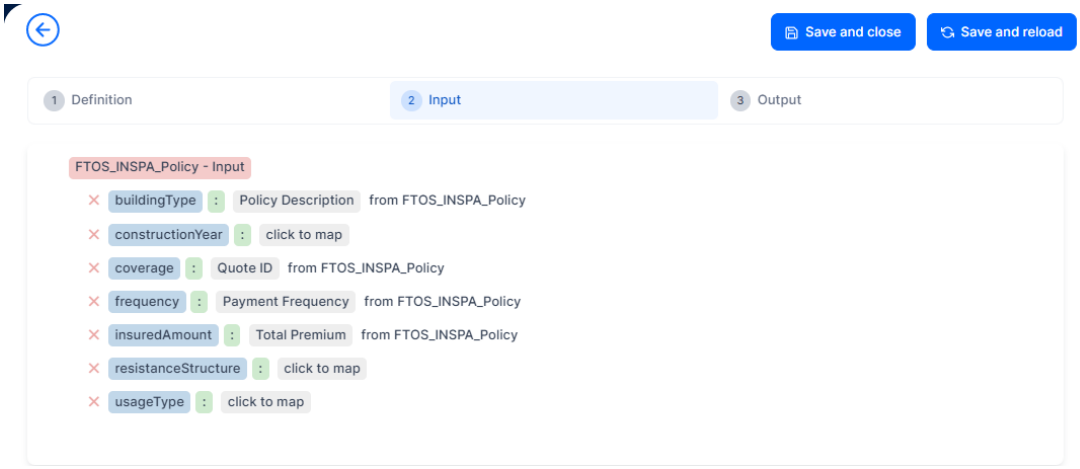
You must have previously selected the **Per Product** underwriting type, in the **Main Info** product tab. This selection makes the **Add Insurance Formula Underwriting** insert form available in the **Underwriting** tab. Changing the **Underwriting Type** option set value, triggers the automatic removal of any formulas previously attached on a product, or coverage.Follow the steps below to add a formula that helps you with scoring:

1. In the Underwriting tab, click Insert. The **Add Insurance Formula Underwriting** insert form is displayed.
2. Insert a **Name** for the formula. The **Insurance Product** field is automatically filled in.
3. In case the **Tariff Type** is set as Per Coverage, select the **Insurance Product Item** to which the formula is applied.
4. Select the **Context Type** for which the formula is applied.
5. Select the formula to be attached for the underwriting calculation.

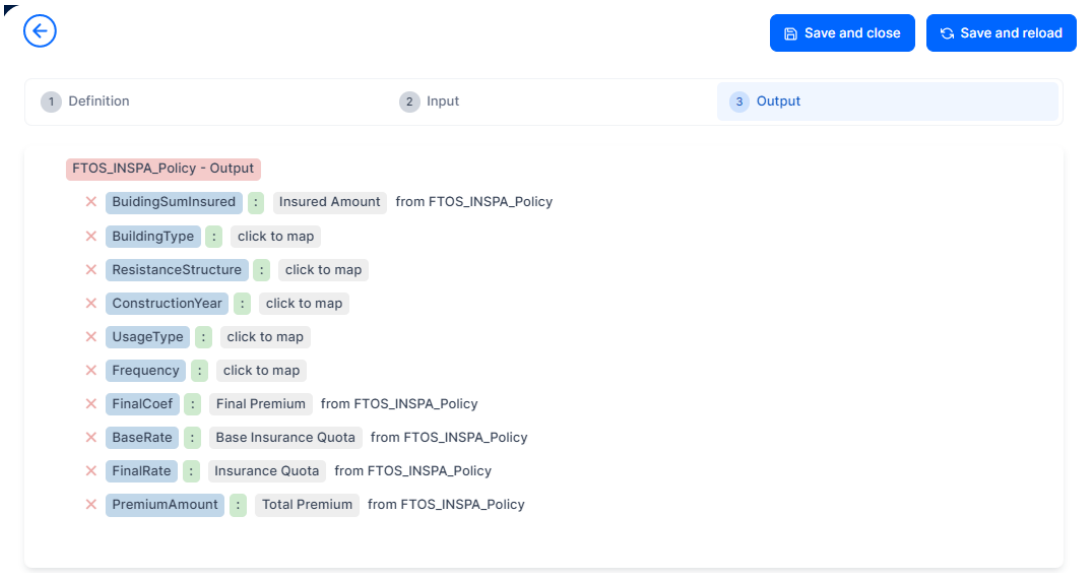


6. Click **Save and Reload**. The **Data Mapping** grid is displayed.
7. Click either **Map Policy Data** or **Map Quote&Bind Data**, depending on the mapping you need to perform. The **Formula Parameter Mapping** form is displayed.

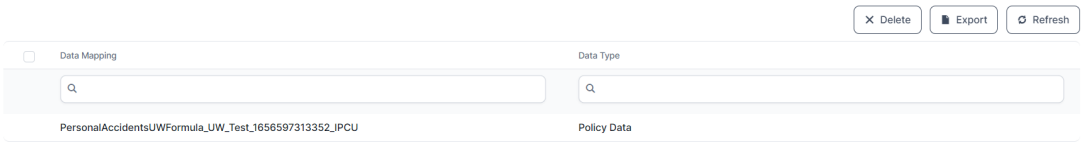
8. In the **Definition** tab, choose a **Master Entity** from which the system extracts the input keys for the formula.
9. Click **Save and Reload**.
10. In the **Input** tab, use the **Click to Map** buttons, next to each **formula input key** (in blue), in order to indicate the attribute (belonging to the previously chosen master entity) from which the system will extract the value needed for calculation.



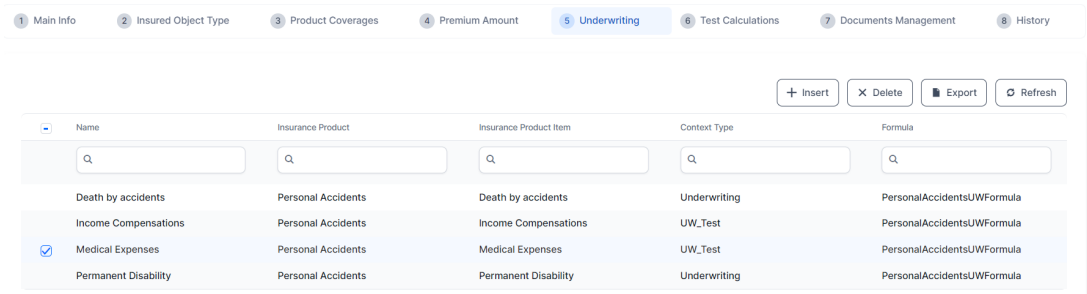
11. Click **Save and Reload**.
12. Go to the **Output** tab, and use the **Click to Map** buttons, next to each **formula output key** (in blue), in order to indicate the attribute to which the system will make updates (insert the value obtained) after the calculation.



13. Click **Save and Close**. You can now see the attached formula in the **Data Mapping** grid.



14. Click **Save and Close**. You can see the Insurance Product Underwriting formula in the Underwriting tab.



15. Continue to tab 6, **Test Calculations**.

Test the Created Formulas

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.

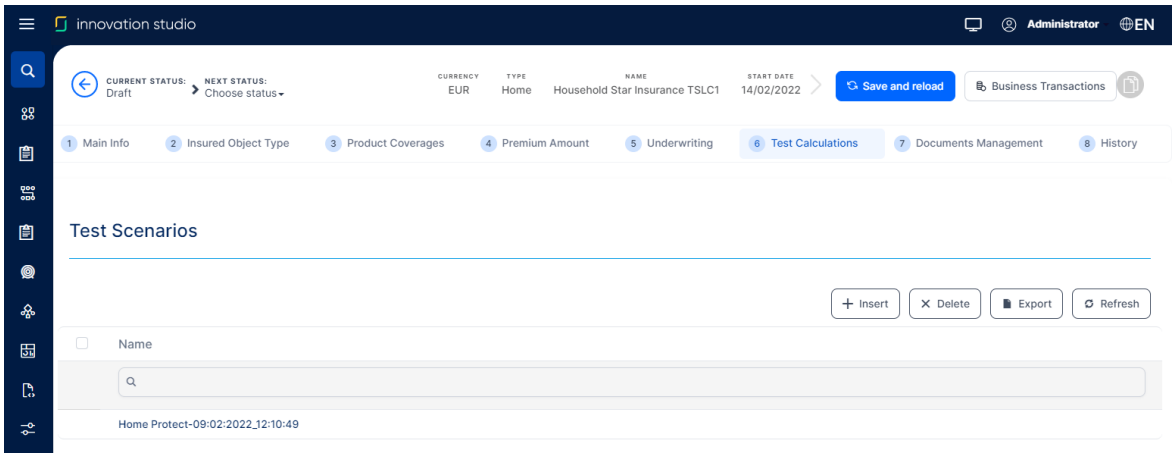
The **Test Calculations** tab allows you to:

- 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.

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

Create A Test Scenario

The **Test Calculations** tab offers you an overview of all the test scenarios registered for the current product.



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 is displayed.

   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**.

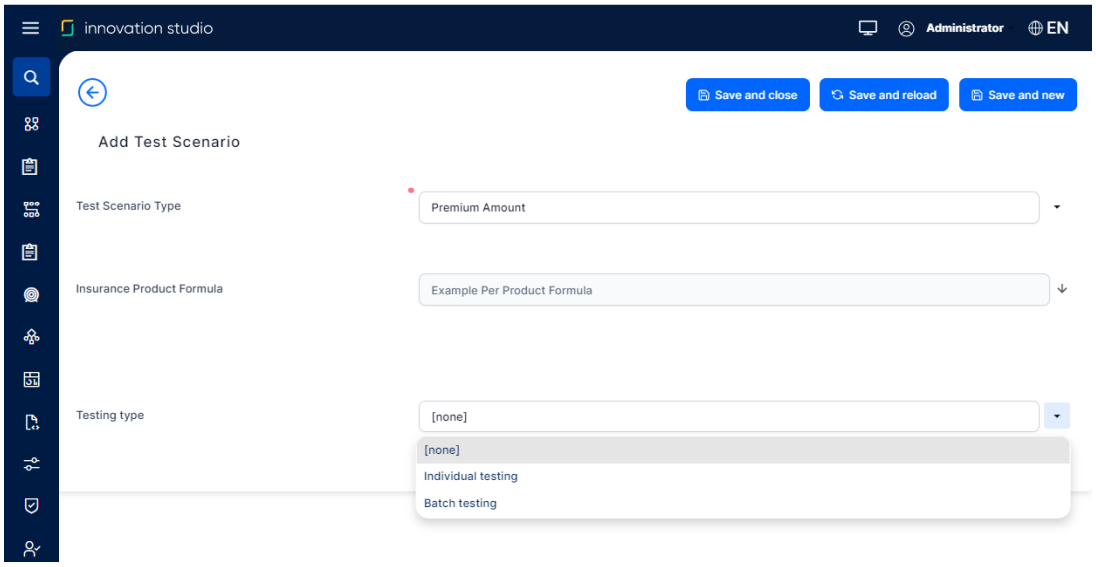
3. Select the **Formula** to be used for calculation:

   - 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,

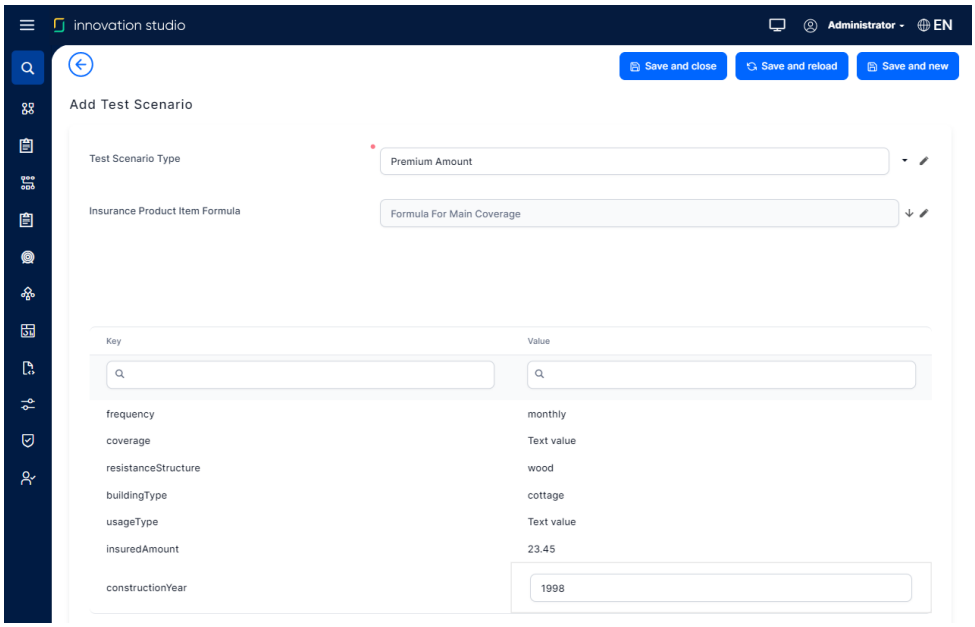
depending on your testing scenario type.



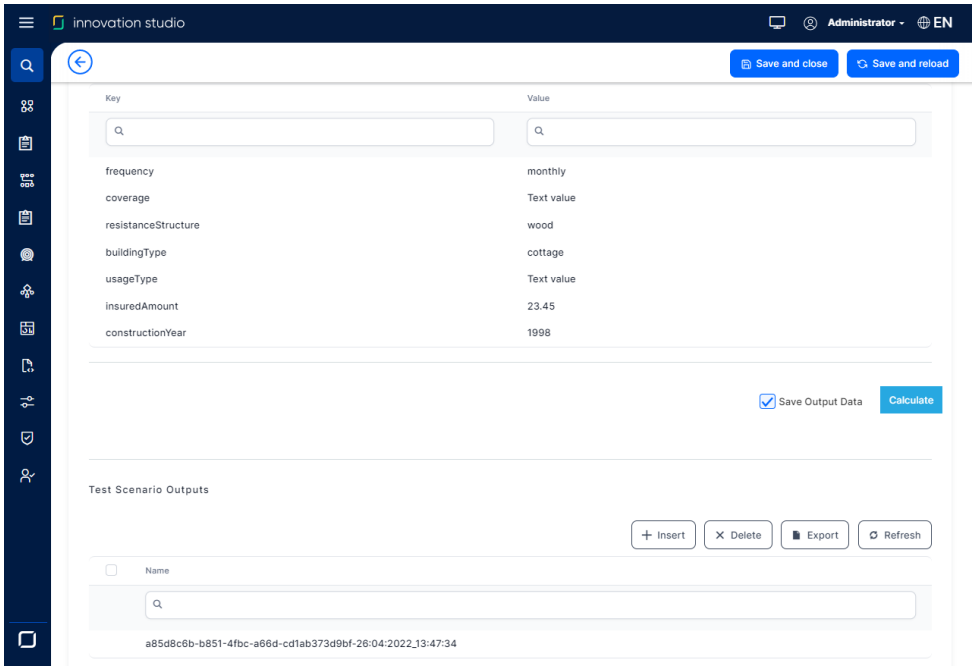
Individual testing

After you have clicked **Save and reload**, the **Key & Value** grid is displayed.

1. Inside this grid, fill in the values for the exposed keys. The text values are case sensitive.



2. Click **Save and reload**. The **Calculate** button and the option to check **Save the Output Data** are displayed. The **Test Scenario Outputs** grid becomes available.



3. Click **Calculate** 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. Click **Save and close**.

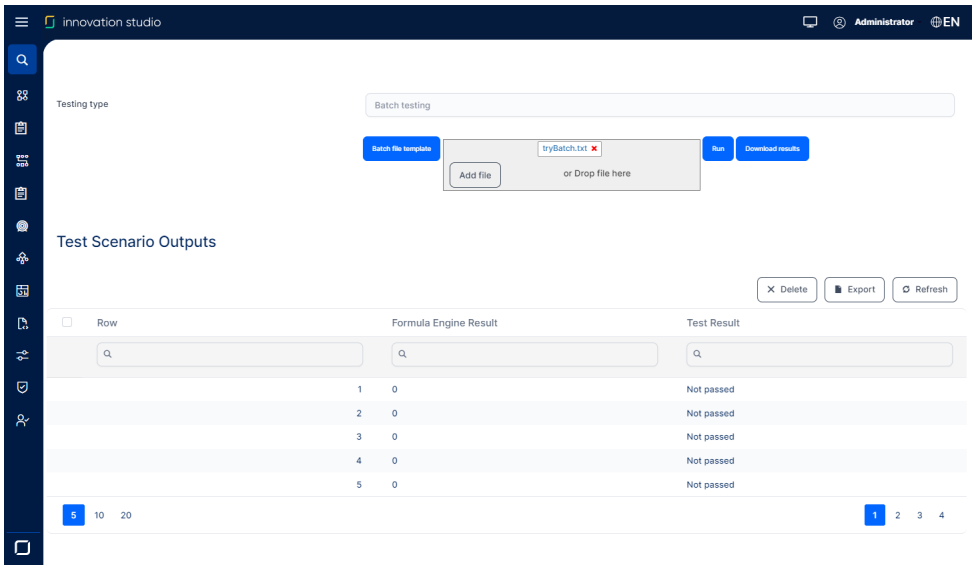
Batch testing

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

1. Click **Batch file template** 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**. 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.



Continue to tab 7, **Documents Management**.

Manage Product Documents

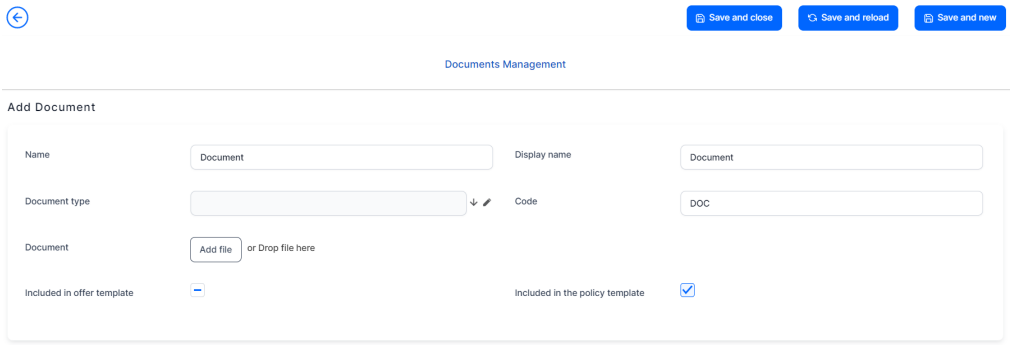
This section allows you to manage the documents that are associated with your product. For example:

- 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.

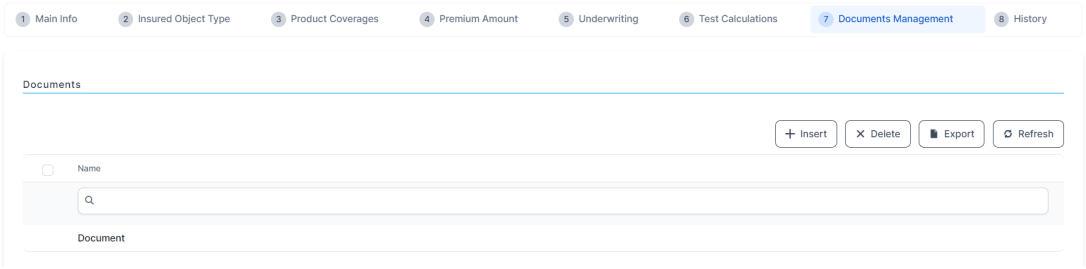
Add Documents

1. In the **Documents Management** tab, click **Insert**. The **Add Document** form is displayed.
2. Fill In the following fields:
    - Name: The name of your document;
    - Display Name: The display name of your document;
    - Document Type: From the drop-down, choose between **Policy**, **Terms & Conditions** or **IPID** - Insurance Product Information Document - which is a necessary type of document 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.



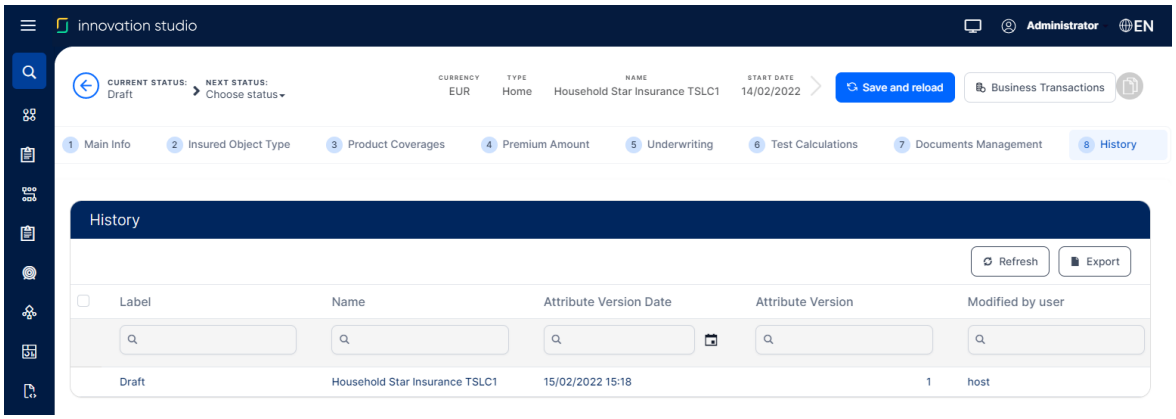
3. Click **Save and Close**. The document record is displayed in the **Documents Management** tab.



4. Continue to tab 8, **History**.

View the Product's 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.



The 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 needs to be in the **Approved** status.

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 **FintechOS 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_CheckUnderwrittingType

Script: FTOS_IP_InsuranceProduct_CheckUnderwrittingType

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.

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.

Get Product 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.

> **NOTE**
> Starting with the **4.6.0** release, this API is compatible with the OpenAPI v3.0 standard.

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
• tariff type - **per product,** • underwriting rules - **per coverage,** • underwriting context type set to **medical underwriting.**	• 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.

```
{
  "validityDate": "2022-12-05",
  "productCode": "PA"
}
```

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:

```
{
  "isSuccess": true,
  "errorMessage": null,
  "errorCode": null,
  "result": [
    {
      "productCode": "PA",
      "tariffType": "perCoverage",
      "uwType": "perCoverage",
      "premiumCalculationStructures": [
        {
          "itemCode": "DPA",
          "formulaVersion": 4,
          "formulaStructure": [
            {
              "key": "sumInsured",
              "value": null,
              "masterType": "SimpleType",
              "subType": "Decimal",
              "objProps": null
```

```
            },
            {
              "key": "frequency",
              "value": null,
              "masterType": "SimpleType",
              "subType": "Text",
              "objProps": null
            },
            {
              "key": "age",
              "value": null,
              "masterType": "SimpleType",
              "subType": "WholeNumber",
              "objProps": null
            },
            {
              "key": "coverage",
              "value": null,
              "masterType": "SimpleType",
              "subType": "Text",
              "objProps": null
            }
          ]
        },
        {
          "itemCode": "ICPA",
          "formulaVersion": 4,
          "formulaStructure": [
            {
              "key": "sumInsured",
              "value": null,
              "masterType": "SimpleType",
              "subType": "Decimal",
              "objProps": null
            },
            {
              "key": "frequency",
              "value": null,
              "masterType": "SimpleType",
              "subType": "Text",
              "objProps": null
            },
            {
              "key": "age",
              "value": null,
```

```
                "masterType": "SimpleType",
                "subType": "WholeNumber",
                "objProps": null
              },
              {
                "key": "coverage",
                "value": null,
                "masterType": "SimpleType",
                "subType": "Text",
                "objProps": null
              }
            ]
          },
          {
            "itemCode": "MEACC",
            "formulaVersion": 4,
            "formulaStructure": [
              {
                "key": "sumInsured",
                "value": null,
                "masterType": "SimpleType",
                "subType": "Decimal",
                "objProps": null
              },
              {
                "key": "frequency",
                "value": null,
                "masterType": "SimpleType",
                "subType": "Text",
                "objProps": null
              },
              {
                "key": "age",
                "value": null,
                "masterType": "SimpleType",
                "subType": "WholeNumber",
                "objProps": null
              },
              {
                "key": "coverage",
                "value": null,
                "masterType": "SimpleType",
                "subType": "Text",
                "objProps": null
              }
```

```
            ]
          },
          {
            "itemCode": "PDA",
            "formulaVersion": 4,
            "formulaStructure": [
              {
                "key": "sumInsured",
                "value": null,
                "masterType": "SimpleType",
                "subType": "Decimal",
                "objProps": null
              },
              {
                "key": "frequency",
                "value": null,
                "masterType": "SimpleType",
                "subType": "Text",
                "objProps": null
              },
              {
                "key": "age",
                "value": null,
                "masterType": "SimpleType",
                "subType": "WholeNumber",
                "objProps": null
              },
              {
                "key": "coverage",
                "value": null,
                "masterType": "SimpleType",
                "subType": "Text",
                "objProps": null
              }
            ]
          }
        ],
        "uwRulesStructures": [
          {
            "contextType": "Underwriting",
            "itemCode": "DPA",
            "formulaVersion": 4,
            "formulaStructure": [
              {
                "key": "age",
```

```
                "value": null,
                "masterType": "SimpleType",
                "subType": "WholeNumber",
                "objProps": null
              }
            ]
          },
          {
            "contextType": "Underwriting",
            "itemCode": "PDA",
            "formulaVersion": 4,
            "formulaStructure": [
              {
                "key": "age",
                "value": null,
                "masterType": "SimpleType",
                "subType": "WholeNumber",
                "objProps": null
              }
            ]
          },
          {
            "contextType": "UW_Test",
            "itemCode": "ICPA",
            "formulaVersion": 4,
            "formulaStructure": [
              {
                "key": "age",
                "value": null,
                "masterType": "SimpleType",
                "subType": "WholeNumber",
                "objProps": null
              }
            ]
          },
          {
            "contextType": "UW_Test",
            "itemCode": "MEACC",
            "formulaVersion": 4,
            "formulaStructure": [
              {
                "key": "age",
                "value": null,
                "masterType": "SimpleType",
                "subType": "WholeNumber",
```

```
                "objProps": null
              }
            ]
          }
        ]
      }
    ]
}
```

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 `perProduct` or `perCoverage`.
uwType	The **Underwriting Type** defined on the product level - either `perProduct` or `perCoverage`.
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.

Code	Text	Description
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 **Get Product Formulas Structures API** 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

Insurance Product APIs

From this library, use the **Get Formula Input Parameters** 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 Underwriting 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).

> **NOTE**
> Starting with the **4.6.0** release, this API is compatible with the OpenAPI v3.0 standard.

The **Get Underwriting Rules Result API** endpoint runs the UW formulas assigned to each product, or product item (coverage), simulating the **Test Scenario** functionality - available for users in **FintechOS Studio**.

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).

All the requests through this API and their responses can be saved into the **Product Interogation History** entity. This action is available if the system parameter **Product Interogation History 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.

```
{
    "productCode": "PETP",
    "validityDate": "2022-12-05",
    "sourceRecordName": "P7856",
    "sourceRecordId": "0AFE8D0B-F423-44F4-9638-
DEE696BF0B0E",
    "uwRulesDetails": [{
        "contextType": "Underwriting",
        "details": {
            "age": 5,
            "petType": "Dog"
        }
    }]
}
```

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 `perProduct` UW type.
`productCode`	The code of the **Insurance Product**.
`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.
`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:

```
{
  "isSuccess": true,
  "errorMessage": null,
  "errorCode": null,
  "result": [
    {
      "contextType": "Underwriting",
      "finalDecision": "Passed",
      "decision": {
        "age": 5,
```

```
            "petType": "Dog",
            "Step1": "Passed",
            "PetProtectUWFormula": "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.

Code	Text	Description
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 **Get Underwriting Rules Result API** 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

Insurance Product APIs

From this library, use the **Get Underwriting Rules Result** 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.

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).

NOTE
Starting with the **4.6.0** release, this API is compatible with the OpenAPI v3.0 standard.

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

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).

All the requests through this API and their responses can be saved into the **Product Interogation History** entity. This action is available if the system parameter **Product Interogation History 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.

```
{
    "insuranceTypeName": "Personal Accidents",
    "productCode": "PA",
    "validityDate": "2022-12-05",
    "sourceRecordName": "P7857",
    "sourceRecordId": "0AFE8D0B-F423-44F4-9638-
DEE696BF0B0E",
    "premiumCalculationDetails": [{
        "itemCode": "DPA",
        "calculationDetails": {
            "age": 30,
            "coverage": "Death by accidents",
            "frequency":"annually",
            "sumInsured":3000
        }
    },{
        "itemCode": "PDA",
        "calculationDetails": {
            "age": 30,
            "coverage": "Permanent Disability",
```

```
            "frequency":"annually",
            "sumInsured":3000
        }
    },
            {
        "itemCode": "ICPA",
        "calculationDetails": {
            "age": 30,
            "coverage": "Income Compensations",
            "frequency":"annually",
            "sumInsured":3000
        }
    },
            {
        "itemCode": "MEACC",
        "calculationDetails": {
            "age": 30,
            "coverage": "Medical Expenses",
            "frequency":"annually",
            "sumInsured":3000
        }
    }
]
}
```

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`.

Parameter	Description
premiumCalculationDetails	Array with details to identify and run the formulas.
productCode	The code of the **Insurance Product**.
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:


```
{
  "isSuccess": true,
  "errorMessage": null,
  "errorCode": null,
  "result": [
    {
      "totalPremiumAmount": 216,
      "tariffType": "perCoverage",
      "premiumCalculationResults": [
        {
          "itemCode": "DPA",
          "premiumAmount": 4.8
        },
        {
          "itemCode": "PDA",
          "premiumAmount": 57.6
        },
        {
          "itemCode": "ICPA",
          "premiumAmount": 96
        },
        {
          "itemCode": "MEACC",
```

```
            "premiumAmount": 57.6
          }
        ],
        "formulaDetails": [
          {
            "itemCode": "DPA",
            "formulaResult": {
              "age": 30,
              "coverage": "Death by accidents",
              "frequency": "annually",
              "sumInsured": 3000,
              "BaseRate": 0.002,
              "CoefFrequency": 0.8,
              "CoefAge": 1,
              "CoefAmountInsuredItem": 1,
              "ItemPremiumAmount": 4.8,
              "PA_Final_Premium": 4.8
            }
          },
          {
            "itemCode": "PDA",
            "formulaResult": {
              "age": 30,
              "coverage": "Permanent Disability",
              "frequency": "annually",
              "sumInsured": 3000,
              "BaseRate": 0.024,
              "CoefFrequency": 0.8,
              "CoefAge": 1,
              "CoefAmountInsuredItem": 1,
              "ItemPremiumAmount": 57.6,
              "PA_Final_Premium": 57.6
            }
          },
          {
            "itemCode": "ICPA",
            "formulaResult": {
              "age": 30,
              "coverage": "Income Compensations",
              "frequency": "annually",
              "sumInsured": 3000,
              "BaseRate": 0.04,
              "CoefFrequency": 0.8,
              "CoefAge": 1,
              "CoefAmountInsuredItem": 1,
```

```
            "ItemPremiumAmount": 96,
            "PA_Final_Premium": 96
          }
        },
        {
          "itemCode": "MEACC",
          "formulaResult": {
            "age": 30,
            "coverage": "Medical Expenses",
            "frequency": "annually",
            "sumInsured": 3000,
            "BaseRate": 0.024,
            "CoefFrequency": 0.8,
            "CoefAge": 1,
            "CoefAmountInsuredItem": 1,
            "ItemPremiumAmount": 57.6,
            "PA_Final_Premium": 57.6
          }
        }
      ]
    }
  ]
}
```

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 `perProduct` or `perCoverage`.
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.
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

Insurance Product APIs

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.

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:

- Get Underwriting Rules Result API,
- Premium Amount API,

- Proposal Configuration Premium Calculation.

Product Interogation History

User Journey

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

Form

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

View

The **Product Interogation History** 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 **FintechOS Studio Product Factory** menu, from menu item **Product Interrogation History**.

System Parameter

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