Accelerate Price List Impact Simulation

Version 1.0.1

November 2023

Accelerate Price List Impact Simulation Package

The Price List Impact Simulation is a visual impact assessment tool that allows Pricing Managers to use updated list prices to simulate the potential impact of the new prices on their business.

Unlike the classical approach, this solution incorporates a holistic way of simulating impact at a granular level (customer x product x time), including special agreement (net prices).

The simulation works with Price Lists and Live Price Grids. As such, in the rest of the document, when something is stated for a PL, the same applies to an LPG.

In this section:

Overview - Price List Impact Simulation

Purpose

Price List Impact Simulation Accelerator intends to provide an easy way to benchmark **impact of list price changes**, using past transactions and a set of hypothesis in order to assess all metrics from the price waterfall, from revenue to profit. It is tied to Price Settings, as the process starts directly from a Price List or a LPG and a model is created to run the **simulation** from the hypothesis set by the user. The results are displayed through a set of **dashboards** and **tables** that can be **reviewed** in details in the model.

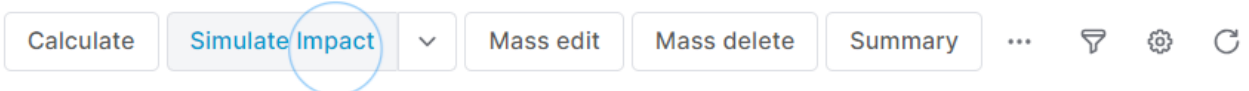
Outputs

The simulation provides a set of tables and dashboards to review the impact of the new list prices coming from Price Settings. The outputs of the simulation stay in the model, they are not pushed back to Price Setting.

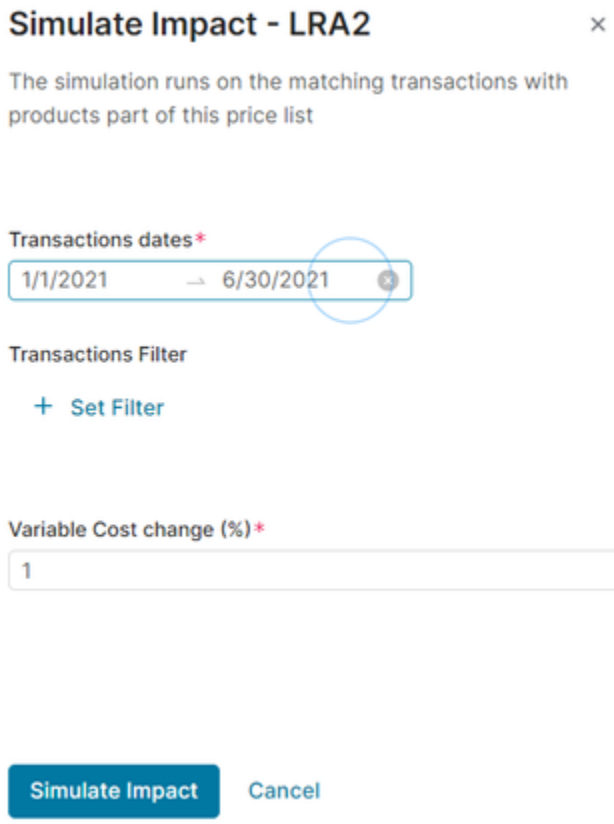
Approach

The Price List Impact Simulation relies on the following steps:

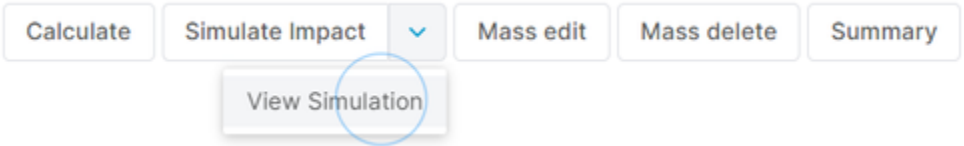
1. Defining List Prices within Price Setting (either in List Prices or LPG).
2. When prices are all good, the simulation can be launched directly from Price Setting with the "Simulate Impact" button (available when the simulation is configured).



3. Hypothesis of the simulation should be then defined, such as the period of transactions to consider in the simulation, the scope of the transactions with the transaction filter (e.g. a specific country, business unit...).



4. The simulation starts and a model is created with the transaction scope and hypothesis. The Optimization Engine computes the simulation and the simulation is made available when all the calculations are over.
5. Dashboards and tables are available in the model, which can be accessed directly from Price Setting.



Limitations

- **Custom waterfall and implementation** - This accelerator relies on a standard waterfall that may differ from a company specific waterfall. So a custom waterfall is possible but requires additional effort, including setting up the Optimization Engine that is used for the simulation and mapping the fields of the dashboards
- **Product scope** - The simulation relies on past transactions as a starting point and a benchmark, so a product with no sales will not be part of the optimization. Also discontinued products may still part of the simulation.
- **No predefined extension point** - There is no out-of-the-box extension point defined for now. If you intend to add specific features, custom code should be written. (But then the accelerator becomes specific and it cannot be updated without extra effort to port those modifications.) Do not hesitate to report specific requirements and possible extension points to Pricefx.
- **Data requirements** - For details on the standard price waterfall see Default Waterfall Description (Optimization - Price List Impact Simulation).

- **Out of scope**:
  - Price Elasticity (may be added with additional customization, but not out-of-the-box)
  - Forecasts of sales

Business User Reference (Optimization - Price List Impact Simulation)

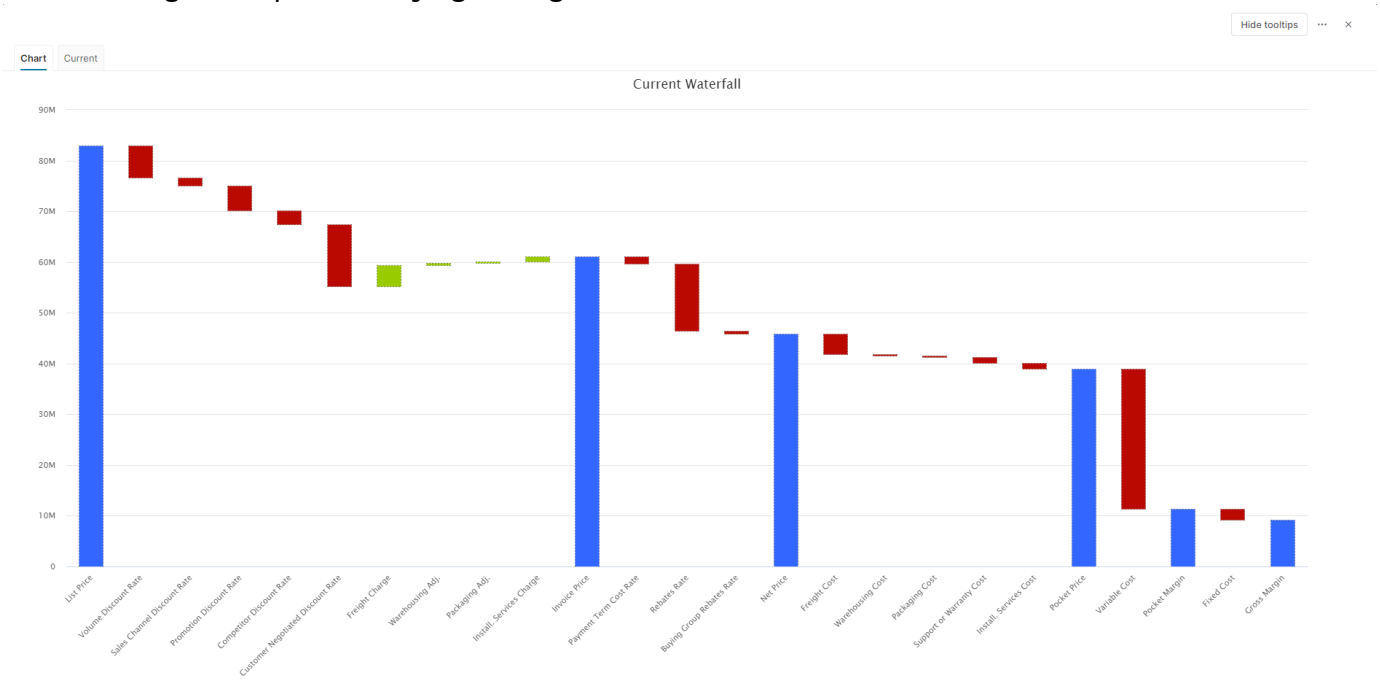
> ⚠ Please be aware that this feature is NOT yet an accelerator and as such, it requires CE work to match a new problem.

The simulation works with Price Lists and Live Price Grids. As such, in the rest of the document, when something is stated for a PL, the same applies to an LPG. Furthermore, it handles secondary keys given that their column names are correctly specified in the Price Setting Type (PLPGTT) configuration.

In this section:

Default Waterfall Description (Optimization - Price List Impact Simulation)

The waterfall simulated by default contains the below listed fields and any changes to their names or their meaning will require modifying the logics.



The waterfall is created from a transactions Datamart that must contain the following fields. All of them contain extended values; the discount and adjustments are *absolute* extended values.

- **List Price** - This field does not need to be present in the Datamart. It is calculated as:

```
FinalBasePrice + (MaterialsIndexFormulaAdj + SegmentMarginAdj +
LocalAdjustment)
```

Therefore it means that the real needed fields are `FinalBasePrice`, `MaterialsIndexFormulaAdj`, `SegmentMarginAdj`, and `LocalAdjustment`. Then the following discounts and adjustments apply to the List Price:

- Volume Discount - name `VolumeDiscounts`
- Sales Channel Discount - name `SalesChannelDiscount`
- Promotion Discount - name `PromotionDiscount`
- Competitor Discount - name `CompetitorDiscount`
- Customer Negotiated Discount - name `CustomerNegotiatedDiscount`
- Freight Charge - `Freight_Charge`
- Warehousing Adjustment - name `WarehousingAdj`
- Packaging Adjustment - name `PackagingAdj`
- Installation Services Customization Charge - name `Services`

- **Invoice Price** - This field does not need to be present in the Datamart. It is calculated as:

```
ListPrice
  - (VolumeDiscounts + SalesChannelDiscount + PromotionDiscount +
CompetitorDiscount + CustomerNegotiatedDiscount)
  + (Freight_Charge + WarehousingAdj + PackagingAdj + Services)
```

Then the following discounts and adjustments apply to the Invoice Price:

- Payment Term Cost - name `EarlyPayment`
- Rebates - name `Rebates`
- Buying Group Rebates - name `BuyingGroupRebates`

- **Net Price** - This field does not need to be present in the Datamart. It is calculated as:

```
InvoicePrice - (EarlyPayment + Rebates + BuyingGroupRebates)
```

Then the following discounts and adjustments apply to the Net Price:

- Freight Cost - name `Freight_Cost`
- Warehousing Cost - name `WarehousingCost`
- Packaging Cost - name `PackagingCost`
- Support of Warranty Cost - name `WarrantyCost`
- Installation Services Cost - name `InstallationCosts`
- Pocket Price - This field does not need to be present in the DM. It is calculated as:

```
NetPrice - (Freight_Cost + WarehousingCost + PackagingCost +
WarrantyCost + InstallationCosts)
```

Then the Variable Cost is applied to the Pocket Price. Its field name is `Cost`

- **Pocket Margin** is calculated as `PocketPrice - Cost`.
  Then the Fixed Cost is applied to the Pocket Margin. Its field name is `FixedCost`
- **Gross Margin** is calculated as `PocketMargin - FixedCost`

The transactions Datamart must also contain a date field.

⚠ In any case, a Configuration Engineer will be needed to at least check that the first aggregation queries run the correct pre-configuration formulas. This is a topic that will be handled by the Accelerator when it exists.

Usage (Optimization - Price List Impact Simulation)

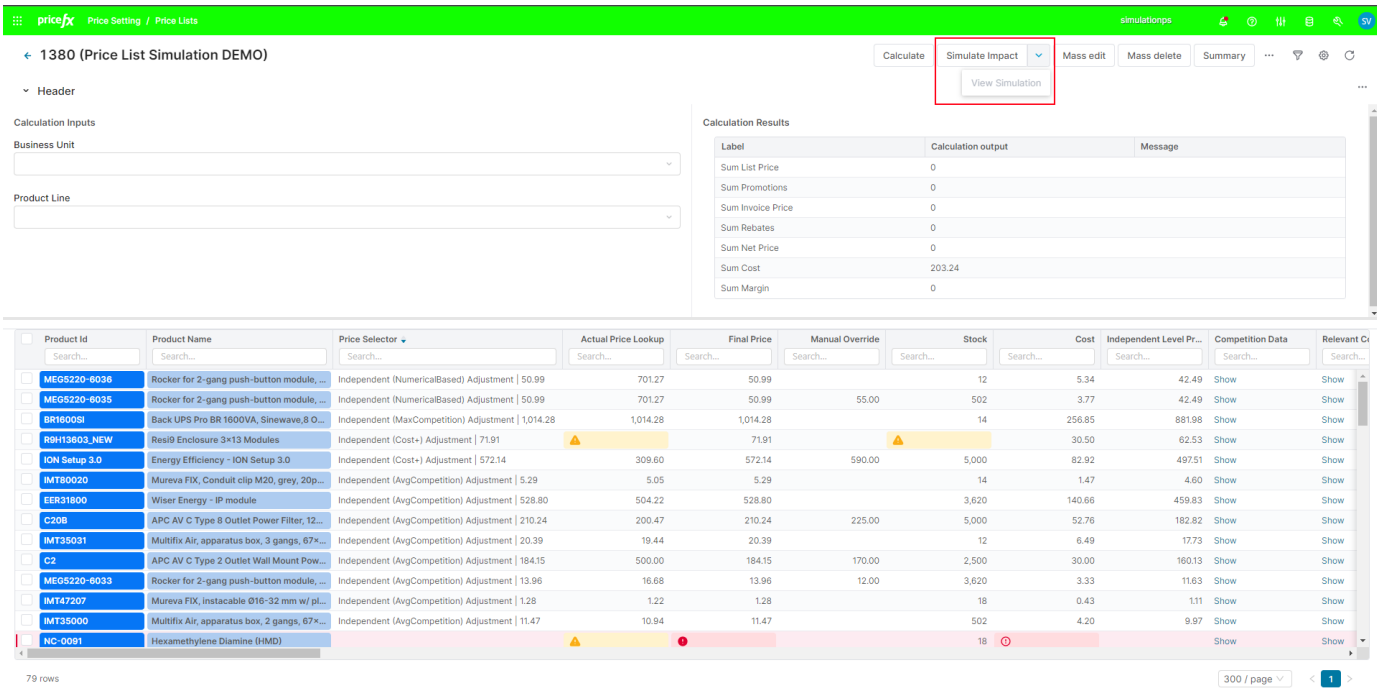
Take the following steps to configure a model. In case of trouble, the last section provides the most common tips.

Step by Step

1. Go to the Price Settings Module

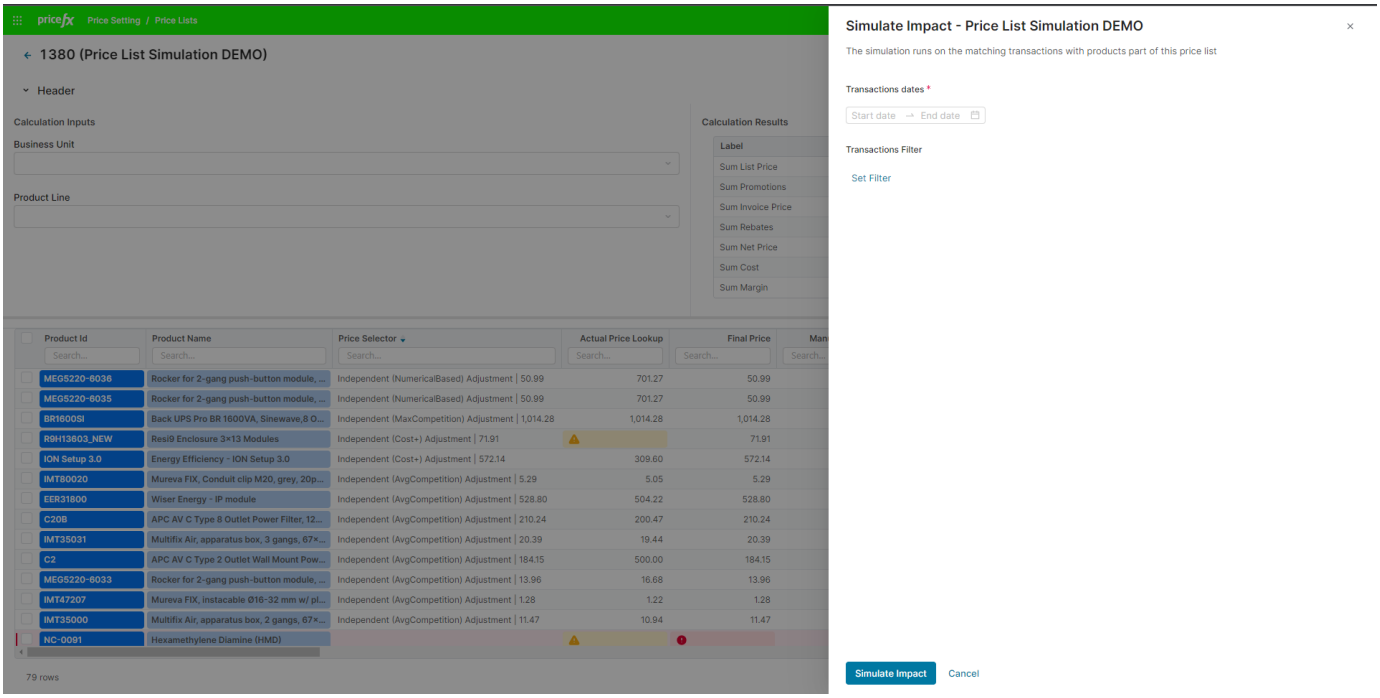
First, you need to open the PL or LPG you want to simulate from the **Price Setting** > **Price Lists** or **Price Setting** > **Live Price Grids** table.

If the configuration of the selected Price Setting Type is correct, the **Simulate Impact** option appears at the top of the Price List. Click it to proceed to the next step.



2. Configure the Simulation

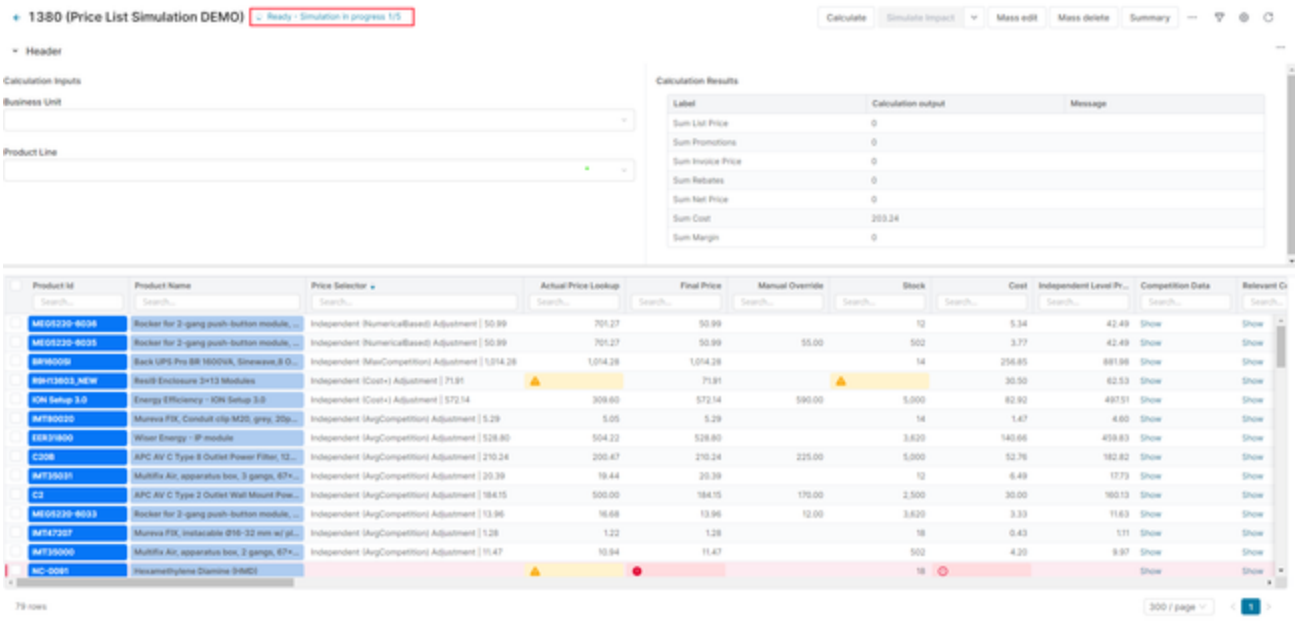
The drawer opens on the right-hand side and presents the options needed to configure the simulation and set the filters.



- The Transaction Dates input defines the time range of transactions to consider during the simulation.
- Only the products having a line in the Price List will be considered for the simulation.

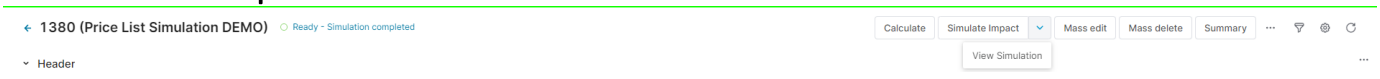
3. Run the Simulation

Once the simulation is started, the Status of the Price List is updated, showing that a simulation is underway and what its current calculation step is. The newly created simulation is named "Simulation of + the name of the PL/LPG".

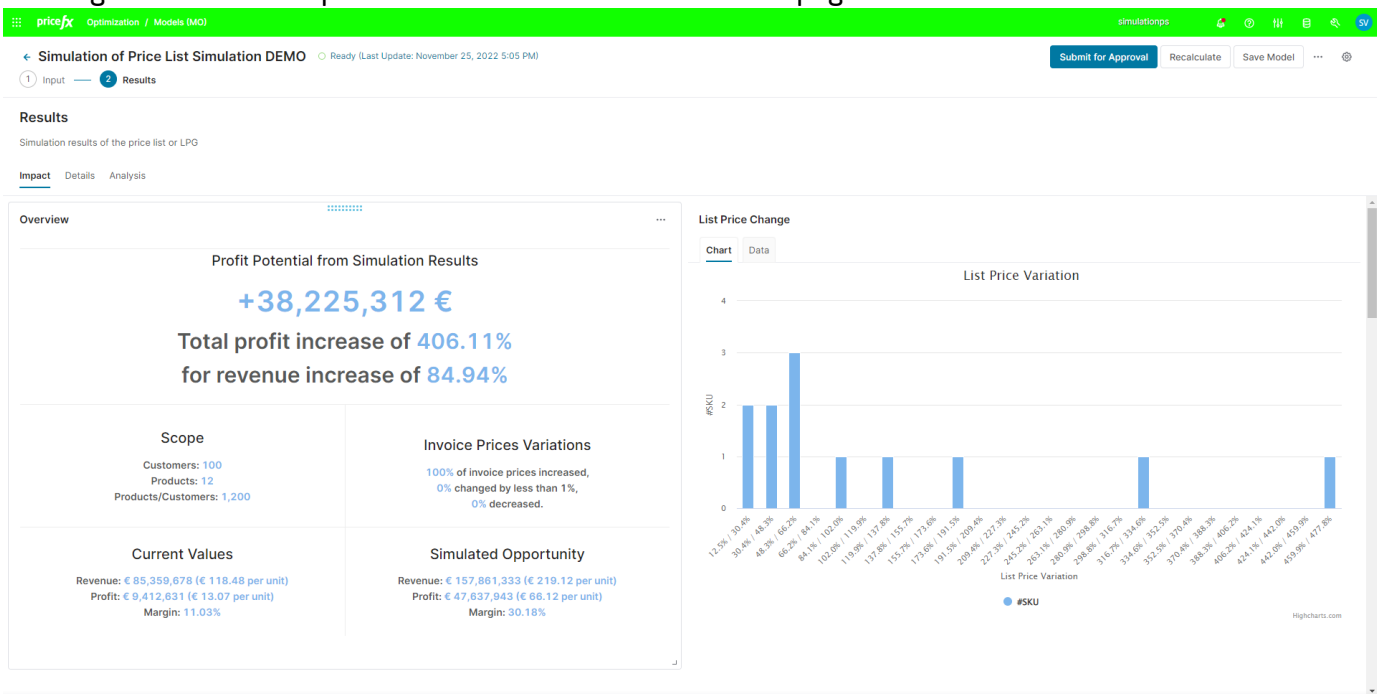


4. View the Results

Once all the steps are executed, the header is updated with the message "Simulation Completed". The results are available by clicking on this status, or selecting the option *View Simulation* from the dropdown under **Simulation Impact**.



Viewing the simulation opens the Model itself and its Results page.



The following three tabs are available:

1. **Impact** - Contains the main comparison charts between the historical data and the simulation results. There are no user inputs in this tab, the charts fill the whole canvas.

2. **Details** - Contains the table view of the results, with a comparison between current and simulated values, aggregated at different levels. Here, we have 3 model tables corresponding to the aggregation by SKU, Customer, and SKU/Customer. This tab also contains the results of the previous Error Check, counting the number of price items ignored due to their incompleteness or missing transactions.
3. **Analysis** - Is a kind of mirror of the Impact tab but with the option to filter the results and therefore, dig deeper to understand how the global view manifests for a given product family or customer, for example.

Note that the results will still be available from the Price List itself by clicking the **View Simulation** button.

Be careful while checking the Model Tables. They are not intended, in their format and content, to be directly readable by the end user. They contain the information needed for the logics. If you want to check some values as a table, do not try to find it in the Model Table but ask a Configuration Engineer for a specific visualization instead, as it is done for the Details tab.

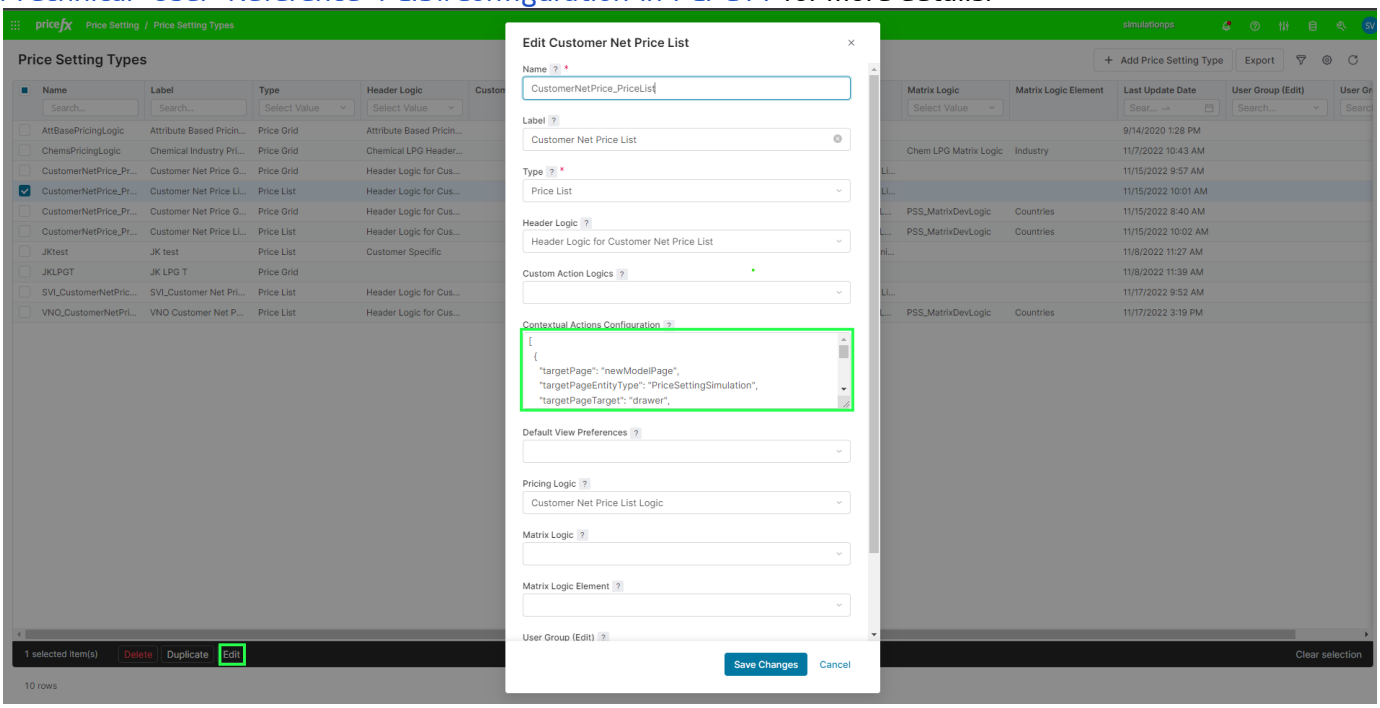
Troubleshooting

This is the list of the most common errors and how to solve them.

Missing the "Simulate" Option

This problem occurs when the Price Setting Type (PLPGTT) configuration is either formatted incorrectly or missing. The first step is to be sure that you are working on a PL/LPG of the correct type. If this is the case, check its configuration on the Price Setting Types page and correct it accordingly.

Refer to https://pricefx.atlassian.net/wiki/spaces/UDEV/pages/4354605149/Technical+User+Reference+PLIS#Configuration-in-PLPGTT for more details.



Error during Calculation with Keywords: "javaheap" "vmem", "memory" or without Message

Lacking the resources to run the model can lead to several errors, depending on the step failing. The most common ones are:

- Error during the Run calculation: This means that the Optimization Engine did not have enough resources. This can be solved by asking the support to increase the RAM allocation for the OE. More information on this issue is available at https://pricefx.atlassian.net/wiki/spaces/UDEV/pages/2812149872/OE+Troubleshooting#Error-during-the-Run-calculation-with-keywords%3A-%E2%80%9Cjavaheap%E2%80%9D-%E2%80%9Cvmem%E2%80%9D%2C-%E2%80%9Cmemory%E2%80%9D.
- Error during the aggregating step: This step basically joins big SQL queries. The size of the transaction DM and of the Price List used can lead to massive queries running out of memory. Unfortunately, the solution here is not as easy as in the previous one and needs to optimize the queries themselves (like breaking the aggregating step into several phases, for example). If these modifications are insufficient, please contact the Pricefx Performance Team (Performance - Team) to improve the most critical queries and operations.

Error during Calculation with Other Messages

The error message in the calculation status should be explicit enough.

The most common case is an error during "CheckData" with a message reading like `"CriticalErrors : No match when looking for transactions. Looked for transactions with products having price, listed in .... Time period for transactions: 2022-12-01 - 2022-12-31. Extra filters: ..."`, meaning that the scope of the simulation is empty. A similar error is thrown when the Price List to simulate is empty.

If the error message does not give enough information, a Configuration Engineer will be able to pinpoint exactly what happened. Note that the majority of these errors come from differences in the data, such as field type changes, empty values, and so on, breaking the existing logics. Use this error to check if the user data need to be cleaned or corrected, or if it is a "feature" of the client data that needs a specific logic modification to be dealt with.

Missing Data in a Price List

Some errors mentioning empty columns or null values often refer to missing data in the Price List. How to solve this issue depends on the project itself.

In some cases, the issue is the format of the Price List itself, which should not allow some fields to be empty and the issue has to be corrected in the Price List.

On the other hand, some situations can prevent changing these Price Lists. The simulation logics themselves can be modified to explicitly check for these errors and deal with them by, for example, skipping the lines missing information in the Price Items logic and logging them.

Admin User Reference (Optimization - Price List Impact Simulation)

Installation (Optimization - Price List Impact Simulation)

Running a Price List Impact Simulation requires at least Pricefx 10.0 (Bee's Knees) and a partition having the following configuration:

- **Optimization Engine (Image)** – To run an optimization model, your partition needs to be allowed to use the Optimization Engine. The Optimization Engine image is automatically enabled on the partition during the accelerator deployment via PlatformManager.
- **Configuration PLPGTT** – A configuration associated with the expected type of PL or LPG must be defined. It describes how to link the PLIS with the PL and how to use its field to autofill some of its inputs. This configuration setting is done in the Price Setting Types view by setting a Contextual Action Configuration. An example of such a configuration is available in the Configuration Engineer documentation Configuration in PLPGTT.
- **Transactions DM** – The fields are currently hardcoded and a Configuration Engineer is needed if the mapping requires any change.

Technical User Reference (Optimization - Price List Impact Simulation)

> This project is not yet an accelerator, meaning that you need to manually deploy the logics and ensure that the Price Setting Type (PLPGTT) configuration is correct. The code is available in the Price Setting Simulation repository: https://gitlab.pricefx.eu/accelerators/price-setting-simulation/

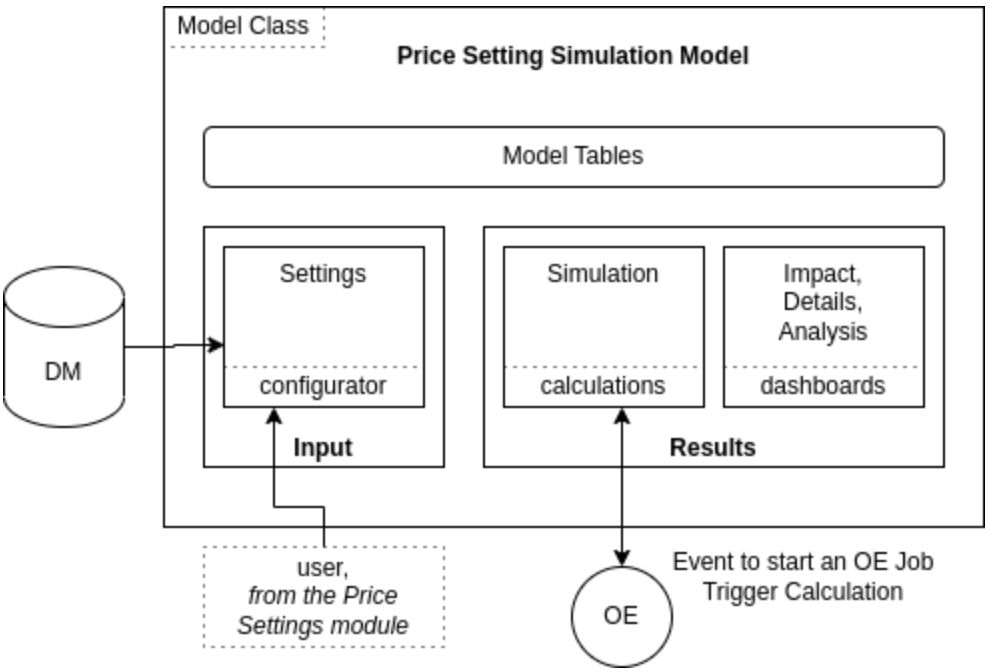
This section details the ModelClass and the logics that define the Price List Impact Simulation feature. For each step, its aim, outputs, and the main reasons to modify the logics are explained.

In this section:

Model Class Structure

The Price Setting Simulation (PSS) Model Class organizes a list of logics to create the model architecture. It is a JSON file that refers to some logics and it is transformed into an optimized UI in the Pricefx platform. See Model Classes for more information.

The general architecture of the Price Setting Simulation Model Class is:



It defines two steps:

- **Input** - Settings of the simulation.
- **Results** - Simulation itself and its results.

There are two types of logics: `calculation`, which writes tables in the model, and `evaluation`, whose purpose is only to display some results. The standard Model Class definition is documented in Model Class (MC).

The logics of the Model Class follow a naming convention. They are all prefixed by `PSS_All_`, then their type is indicated by `Eval` or `Calc`. The configurations are suffixed by `_Configurator`.

Library

The logic is **PSS_Lib**.

Aim of the logic

This library contains the definition of the various columns, names, Datamart, etc. manipulated by the logics.

Common reasons to modify the logic

It is the first place where to rename the source Datamart and the fields used by the price simulation. But as this Model Class is the first step toward a generic definition of the simulation problem and its use in an accelerator, therefore some names referring to the current use case are still hardcoded in the logics. Don't forget to check elsewhere in the other logics too.

Input Step

There is no calculation logic in this step, and there is one tab with related configurator logic **PSS_All_Eval_Price_List_Configurator**.

Aim of the logic

This logic allows the user to enter a set of filtering options that will be applied to the simulation. These inputs are defined in a single configurator which is displayed in the drawer when we start a simulation from a Price List.

- Outputs of the evaluation
  This is the place from where the date range and the custom filters defined by the user are provided to the rest of the model.

  But this logic also determines the type of the source (either a Price List or a Live Price Grid) and its ID.

  Later in the logics, you will access those outputs with the command `model.inputs("input", "price_list")['output key']`.

- Common reasons to modify the logic
  The most common reason is to **change the inputs in the drawer**. Be careful not to use several tabs as only the first one will be rendered in the Price Setting module.

  Also, keep in mind that this configurator must work from the drawer tab as well as from the Models view. The current coding of the Price List input gives an example of how to deal with this requirement with inputs initialized from configuration or awaiting input from the user depending on the use case.

Results Step

This step performs first a sequence of calculations:

- The aggregation of the data is done by **PSS_All_Calc_Aggregating**.
- The data is checked by **PSS_All_Calc_Check_Data**.
- The tables that are required to instantiate the Optimization Engine are stored by the logic **PSS_All_Calc_Store_Problem**.
- The Optimization Engine is set and triggered by **PSS_All_Calc_Run_Simulation**.
- After its run, some postprocessing tasks are done to provide good inputs to the dashboards, in **PSS_All_Calc_Prepare_Results**.

Then, three tabs are displayed, based on these logics:

- The Impact tab is based on **PSS_All_Eval_Impact**.
- The Details tab is based on **PSS_All_Eval_Details**.
- The Analysis tab is based on **PSS_All_Eval_Analysis** and **PSS_All_Eval_Analysis_Configurator**.

Calculation: Aggregating

The logic is **PSS_All_Calc_Aggregating**.

- Aim of the logic
  This logic aggregates the data needed for the simulation and applies the user-defined filters.

- Outputs of the evaluation
  The `PriceItems` table contains the information read from the PL/LPG and the `AggregatedData` table contains the current values of the waterfall computed directly from the historical transactions.

  The `PriceItems` table is used to provide a reference price to the aggregated data. The `AggregatedData` table will be used as a basis of comparison for the Simulation in the charts.

- Common reasons to modify the logic
  Waterfall structure

The aggregated data is built using the waterfall definition provided in Default Waterfall Description (Optimization - Price List Impact Simulation). If you want to change the waterfall structure or the field names, it is the first place to do it.

Price Items join

This aggregation is based on a hidden secondary key, which means that the Price Items are not joined to the transactions Datamart only by the product IDs, but also by this secondary key. You may remove it or change its corresponding field.

Aggregation period

If you want to change the aggregation period:

- If the period is a field of the Datamart, the logic reads the field `PricingDateMonth` from the transaction Datamart. Change this reference.
- If a more complex aggregation must be done, then the queries must be updated.

Calculation: Check Data

The logic is **PSS_All_Calc_Check_Data**.

Aim of the logic

This logic checks if the input data contain blocking or non-blocking errors.

Outputs of the evaluation

The critical errors will throw an error message to the job tracker and stop the calculation sequence. The critical errors are an empty PL/LPG or a PL/LPG where none of the price items matches the transactions Datamart.

The non-critical errors will return a list of non-critical errors: the list of the price items which do not have a price in the LPG/PL and the list of the price items in the PL/LPG that do not correspond to any transaction in the transactions Datamart.

These outputs are accessible with the command `model.outputs('results', 'checkData')`.

Common reasons to modify the logic

You may want to add or remove some errors. The main point is that the critical errors stop the calculation process, and the non-critical ones will just provide a piece of information.

Calculation: Store Problem Tables

The logic is **PSS_All_Calc_Store_Problem**.

Aim of the logic

This logic extracts data and formats them in a way understandable by the Optimization Engine. More details in Problem Tables. There is strict formatting for these tables.

Outputs of the evaluation

The data manipulation creates the model tables prefixed by *Problem_* that act as an endpoint for the OE. Be careful, *their names follow a strict format*: These endpoints must be named according to the `Problem_nameOfTheSpace_nameOfTheScope` present in the *ProblemDescription.groovy* and return the corresponding data. In this Model Class, we have only one scope, so the name on the table should stay `Problem_BySKUCustomerAndPeriod_All`.

Common reasons to modify the logic

Plumbing

If there are different data to send to the OE, it is the place where to write them. The fields SKU, `customer` and `period` are required by the space definition. All the other ones depend on the problem definition: Did the waterfall definition change? Did other pieces of the problem change? Depending on the modification done to the Problem Description File (in Calculation: Run Simulation), you may need to read new inputs from the data to instantiate the new parts of the model. These new fields must be added to the problem table. The way the model was changed will define where to add their name, space and scope in the Problem Description File.

For example, if we add a variable `new_field_to_read` in the scope `all` of the space `BySKUCustomerAndPeriod`.

```
[       "name": "NewVariable",
        "type": "static",
        "init": [
                "type" : "data",
                "field": "new_field_to_read"
                ],
]
```

We need to add this information in the model table `Problem_BySKUCustomerAndPeriod_All`, which means modifying the element `Problem_BySKUCustomerAndPeriod_All` in this logic to add the field `new_field_to_read` to it. For more information see Problem Tables and Problem Description.

If the new data is not available and requires for example reading new fields from the Datamart, then the Aggregation logic (Calculation: Aggregating) must also be updated accordingly and the new fields added to the `AggregatedData` table.

Calculation: Run Simulation

The logic is **PSS_All_Calc_Run_Simulation**.

- Aim of the logic

  This logic contains the Problem Description File and the code to trigger an Optimization Engine. Refer to Problem Description for more details. It runs a simulation on the OE.

- Outputs of the evaluation

  The successful execution of the OE creates several model tables prefixed by `Simulation_*` and containing the output of the simulation. A simulation table is created for each computed variable of the problem description and its name follows a strict convention: `Simulation_<name of the space>_<name of the variable>`.

  The exposed computed variables are `listPriceAfterRates`, `InvoicePrice`, `NetPrice`, `PocketPrice`, `PocketMargin` and `GrossMargin`.

- Common reasons to modify the logic

  For a deep explanation how to write a Problem Description File, refer to Problem Description.

  The main hypothesis of the current model is that everything is modeled at the finest granularity level. It is recommended to keep the modifications at the same level, even if suboptimal, to facilitate the different aggregation steps. In the existing code, the level is SKU / Customer Id / Period.

Changing the Problem Description is the first step as it will impact the variables used and therefore what we need to read and where. It may imply these changes:

- What data to read;
- Where to read them;
- How to use them in the model;
- Does it change the results shown to the user.

Calculation: Prepare Results

The logic is **PSS_All_Calc_Prepare_Results**.

Aim of the logic

This logic formats the raw Optimization Engine output in a usable way to display relevant information to the user.

Outputs of the evaluation

Four tables are created:

- `Simulated` aggregates the output tables of the OE in one table. Its structure is similar to the `AggregatedData` one. The difference is that the "base" price is not read from historical data anymore but from the Price List.
- `Details_Customer` provides the values at a customer and product group aggregation level.
- `Details_Product` provides the values at a product and business unit aggregation level.
- `Details_ProductCustomer` provides the values at a product and customer aggregation level.

The values provided by the Details tables are always the extended and the unit list price, the extended and unit gross margin, and the gross margin rate. For each of these values, the current (coming from the `AggregatedData` table) and the simulated (coming from the `Simulated` table) values are calculated, plus the delta and the delta rate.

Common reasons to modify the logic

In the problem description, if you add or remove a variable, you will have to reflect the change in the `Simulated` table.

The `Details` tables are displayed in the Results step, Details tab. They have to be changed if the user wants to see different fields.

You may also add a new table here if there is a new chart or a new table to display in the Results step.

Impact Tab

The logic is **PSS_All_Eval_Impact**.

Aim of the logic

This logic builds the main comparison charts between the historical data and the simulation results. This is the view the user encounters once the simulation is executed. There are no user inputs in this tab, the charts fill the whole canvas.

Outputs of the evaluation

This logic creates a dashboard made of seven portlets:

- *Overview* – Summary of the changes when applying the prices from the Price List to the transactions.
- *List Price Change* – Histogram displaying how many product prices are different between the historical and the simulation results, depending on the relative price variation.

- *Margin & Revenue by Period* - Bar chart comparing the historical and the simulated values of the total gross margin and the total revenue by a period of time.
- *Current Waterfall, Simulated Waterfall*, and *Waterfall - Current vs Simulated* - Historical waterfall, the simulated one, and a comparison between them.
- *Waterfall Comparison (Current List Price as base 100)* - Relative comparison between the waterfall.

Common reasons to modify the logic

I want to add Results Charts

Let's say we added a new discount in the waterfall, we now must change the existing charts to display this new information.

However, as we can only create tables in a calculation logic, some new charts may need to add an element to store the data in a usable fashion. It should be done in `PSS_All_Calc_Prepare_Results` (Calculation: Prepare Results).

Note that this can lead to slight differences in the implementation of the same chart between the Impact or Details tabs and the Analysis one as the Analysis tab must be updated given the set of filters inputted by the user. For example, the tables in Details are displayed using `dmCtx.buildQuery(query)` which is efficient but not usable for the tables in Analysis using a `toResultMatrix()` as their filtering relies on SQL query.

Details Tab

The logic is **PSS_All_Eval_Details**.

Aim of the logic

This logic builds the table view of the comparison of the current and the simulated values. They are aggregated at different levels. This tab also contains the results of the previous Error Check.

Outputs of the evaluation

The dashboard displays three model tables corresponding to the aggregation by product, customer, and product/customer. These tables are built in Calculation: Prepare Results.

The last portlet provides a summary of the non-blocking errors, calculated in Calculation: Check Data.

Common reasons to modify the logic

If another table is created and should be provided to the user, this is the place where to add it.

If the non-blocking errors are defined differently, it would be better to change the wording here.

Analysis Tab

The logics are **PSS_All_Eval_Analysis** and **PSS_All_Eval_Analysis_Configurator**.

Aim of the logic

This logic is a kind of mirror of the Impact tab but with the option for the user to filter the results and therefore, dig deeper to understand how the global view manifests for any specific product family or customer.

Outputs of the evaluation

The configurator logic `PSS_All_Eval_Analysis_Configurator` provides the user inputs: options parameters to filter the model tables. The main logic then provides the same portlets as the Impact tab Impact Tab but only in the scope that the user has defined.

Common reasons to modify the logic

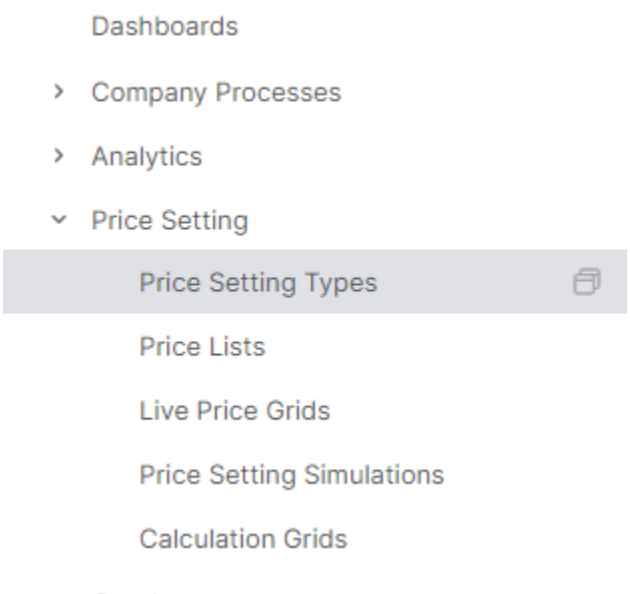
Let's assume that you make similar changes in this tab and in the Impact one.

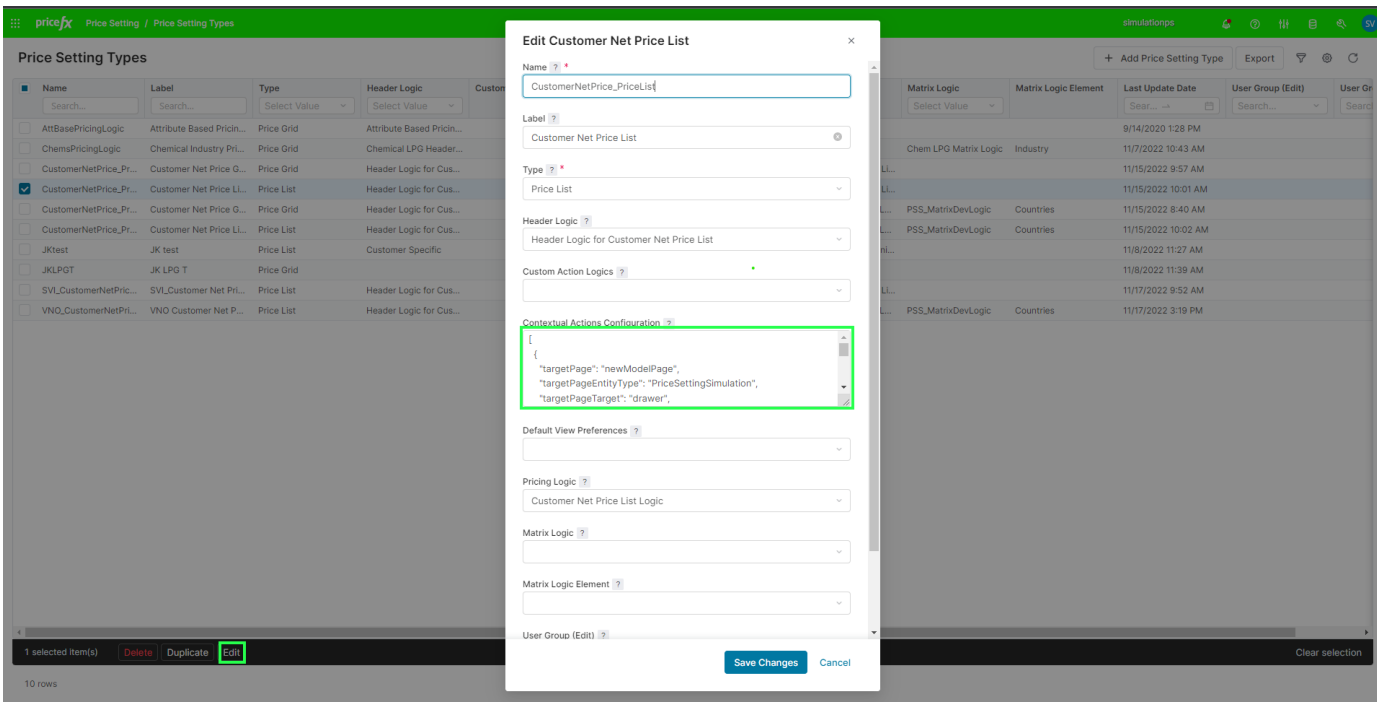
Because of the scope filtering, note that this can lead to slight differences in the implementation of the same chart between the Impact or Details tabs and the Analysis one. For example, the tables in Details are displayed using `dmCtx.buildQuery(query)` which is efficient but not usable for the tables in Analysis using a `toResultMatrix()` as their filtering relies on SQL query.

It is also possible to define other user inputs to filter the scope, for instance numerical filter on the revenue or on the margin.

Configuration in PLPGTT

In order to enable the simulation for a PL/LPG, we need to link them together. This is done by setting a specific "Contextual Actions Configuration" in the correct Price Setting Type.





The field is named `contextualActions` and is a JSON following this model:

```
[
  {
    "targetPage": "newModelPage",
    "targetPageEntityType": "PriceSettingSimulation",
    "targetPageTarget": "drawer",
    "targetPageInputs": {
      "input": {
        "price_list": {
          "SourceTypedId": "{typedId}",
          "secondaryKeyColumnName": "Country"
        }
      }
    }
  }
]
```

It defines:

- The Model Class to use for the simulation in the `targetPageEntityType`.
- The view type of its first step for configuration from the PL/LPG list in the `targetPageTarget` - here it is a `"drawer"`.
- Matching between the PL information and the automatic setting of the inputs in the drawer in the `targetPageInputs`. Here we set the inputs `SourceTypedId` and `secondaryKeyColumnName` of the tab `price_list`, which is the only tab of the first step of the Model Class.

Creation from a Model Object List

Finally, note that the Simulation can be instantiated from the Model Objects list, allowing the creation of several simulations per a Price List and making the development and debugging easier. To do so, simply create a new Model Object using the correct Model Class (more details in Models). The first step will then allow configuring the input to use for the simulation (PL, LPG and optional secondary keys). Be sure that when you add inputs in the first step, they are both usable from the Models view and from the drawer in the PL/LPG simulation option and auto-filled with the PL data when needed. For that two actions are needed:

- Updating the logic building the user inputs (refer to Input Step).
- Updating the mapping in the drawer (refer to the previous paragraph Configuration in PLPGTT).

Release Notes (Optimization - Price List Impact Simulation)

- Optimization - Price List Impact Simulation 1.0.1

Optimization - Price List Impact Simulation 1.0.1

This document summarizes major improvements and fixes introduced in the Price List Impact Simulation Package release version.

Version	1.0.1
Release Date	Nov 30, 2023

New Features and Improvements

New Feature Description	ID
The Optimization Engine image is now automatically enabled on the partition when you deploy the accelerator using PlatformManager.	PFPCS-7331
The accelerator uses now Optimization Engine v2.	PFPCS-7636

Fixed Issues

Description	ID
The postprocessing step Detail_Customer does not work if there are multiple customers.	PFPCS-7332