



Pricefx Accelerators  
Price Setting Package 2.1.0  
January 2022

# Accelerate Price Setting Package (PSP)

With the help of Price Setting Package you can manage Price Lists / Live Price Grids and set up your pricing processes.

- [Product Info \(PSP\)](#)
- [Price Setting Business Introduction](#)
- [Price Setting Fundamentals](#)
- [Price Setting Modules](#)
- [Price Setting Administration](#)
- [Price Setting Technical Information](#)
- [Price Setting Webinars and Other Materials](#)
- [Price Setting Versions](#)

## Product Info (PSP)

**Accelerate Price Setting Package** (in close cooperation with **Accelerate Price Flexibility Package**) provides a price setting and management framework powered by eight distinct pricing strategies that pricing managers can use to create list prices, applying different strategies to different parts of their product portfolio as appropriate. Price Lists can be easily assigned to customers within their markets.

This package includes:

- Gross & Net Price Lists
- Standalone or independent and dependent Price Lists (e.g., National & State or Global & Country, etc.)
- Dynamic pricing (e.g., automated price changes based on an underlying product or competitor changes)
- Eight distinct pricing strategies like attribute-based pricing or competitive based pricing
- Approval workflow through the Accelerate Approval Workflow Package

This is a game-changer. We give pricing managers a framework that allows them to create new list prices quickly, which they can then communicate to their sales teams or customers immediately.

Label	Valid After	Industry	Business Unit	Product Group	Price Strategy #1	Price Strategy #2	Price Strategy #3	Price Strategy #4	Price Strategy #5	Prioritize Independently
USER: 102 Pricing Exceptions (PS)	1/1/2021									
USER: 100 Price strategy selection (PS)	1/1/2021									
USER: 101 Minimum margin guardrails (PS)	1/1/2021									
USER: 101 Minimum margin guardrails (PS)	7/10/2020									
Industry: Chemicals (4)										
Chemicals	*				AvgCompetition	MinCompetition	PriceIncrease	Cost+		
Chemicals	Intermediates			Commodity	AvgCompetition	MinCompetition	PriceIncrease	Cost+		
Chemicals	Resin			Commodity	AvgCompetition	MinCompetition	PriceIncrease	Cost+		
Chemicals	Resin			Specialty	MaxCompetition	AttributeBased	Cost+	AvgCompetition		
Industry: Discrete manufacturing (7)										
Discrete ...	*				Cost+	PriceIncrease	Kit	AvgCompetition		
Discrete ...	Residential and Small Business			Electrical Protection ...	RBP	Cost+	MaxCompetition	NumericalBased		
Discrete ...	Residential and Small Business			Home Automation	MaxCompetition	Cost+	AvgCompetition	AttributeBased	NumericalBased	
Discrete ...	Residential and Small Business			Light Switches and E...	Anchor	Cost+	MinCompetition	AttributeBasedHeader		
Discrete ...	Residential and Small Business			Network Infrastructu...	Kit	AvgCompetition	Cost+			
Discrete ...	Residential and Small Business			Services	MinCompetition	Cost+				
Discrete ...	Residential and Small Business			Uninterruptible Pow...	PriceIncrease	AvgCompetition	Cost+			

## Price Setting Business Introduction

The Price Setting Package helps you manage Price Lists and Live Price Grids through a variety of built-in tools. These include:

- Management of **independent and dependent levels** for price lists. You can define an independent "global" price list and then define dependent "country" price lists that will have prices based on the "global" one. The global country approach is only one possible scenario, you can also build any use case where you have different pricing levels with dependency connections between them.
- Support for various **trade levels**. This allows you to define whether prices are calculated as List Prices or as Gross List Prices and then, after discounts are applied, as Net Prices. Usually List Prices are built in B2C environments where you can typically find Net Prices as discounted List Prices in B2B (B2C) environments where you may want to calculate some List Price and some discounted Net Price for your next trade level.
- The package employs several **price strategies** which you can choose from, such as calculations based on anchor, competition, BoM data or attribute-based pricing.
  - There are extra setup options for these strategies, i.e. you can define their hierarchy, on which level they are valid, whether they can be overridden, how they work with exceptions.
- Product Segmentation helps you define all your pricing decisions on every level of granularity. You can set up dimensions of your product master data that are used to diversify your pricing strategy. The dimensions are looked up hierarchically, so that you can define some high-level pricing strategy and specify special product segments on a more granular level where needed.
- The package can be easily customized through **modularization**. Individual features are prepared as modules which can be turned on or off. Each module can also be customized. The modules cover the following functionality:
  - Through **conditions** you can decide that some prices will be ignored or taken with lower priority. So you can „finetune“ your decision tree for the correct proposed price or ensure general rules in your pricing.
  - You can have **sales and forecast data** displayed. They are usually taken from your sales transactions. You can also deliver sku-aggregated data.

- You can create custom price behavior by creating **exceptions** and allowing **manual override**
- You can have an automatic **price check** whether the margin is within a suitable range.
- You can **round** prices to business friendly values.
- You can choose among different types of **cost calculations** (single, average, sum).
- You can run the PL/LPG calculation with a **volume breakdown** which allows you to apply different volume discounts, depending on the item quantity.
- In addition, the package is well prepared for handling **errors** and issuing **warnings** if the calculations fail.

## Price Setting Use Cases

The Price Setting Package can be used both for Price Lists and Live Price Grids (LPG). The use cases are based on either price levels (one price list or more dependent price lists) or trade levels (list/gross/net prices).

- [Price Levels Scenarios](#)
- [Trade Levels Scenarios](#)
- [Technical Notes](#)

### Price Levels Scenarios

- **Standalone Price List** - Covers price setting for just one price list.
- **Independent Price List / Dependent Price List** - Covers price setting on the independent/dependent levels. Independent price lists are a base for dependent price lists; this allows to create various pricing scenarios, for example: define an independent "global" price list and then define dependent "country" price lists that will have prices based on the "global" one.
- **Complex pricing level structure** - You can build a complex structure. You can define several **Independent Price Lists** calculating prices independently. You can arrange various **Dependent Price Lists** around them. You can also create a multi-level tree where a Dependent Price List depends on another Dependent Price List. With this you can have for example multiple Independent Pricelists for different regions (EMEA, AMER, APAC). Dependent on them you can have Price Lists for the different countries, and dependent on them different channels, shops, ...

### Trade Levels Scenarios

- **List Prices** - Prices are calculated as List Prices.
- **Gross List Price / Net Price** - First, prices are calculated as Gross List Prices. Depending on a Discount Group, a corresponding discount is applied for calculation of the Net Prices.

### Technical Notes

- Also, the [Price Flexibility Package](#) is fully compatible to "monitor" independent LPGs.
- The groups can be defined in the DependencyConfiguration PP and selected during PL/LPG creation using the configurator.

## Volume Breakdown

It is possible to run a PL/LPG with a volume breakdown. This feature allows you to apply different configurations per volume, depending on the quantity. To use this feature, utilize the secondary key in Matrix PL/LPG. The secondary key set will be the list of volumes.

Also, a calculation item in the PL/LPG with the volume equaling to 1 will always be added. It is to keep the default price without the volume discount applied and it will be used as the independent item price in dependent PL/LPG calculations.

#### Discount

The calculation logic takes the secondary key, looks for the additional discount, and applies the adjustment to list prices. There will be the `Volume Discount` element to show the discount per volume value. If there is no volume breakdown defined for an SKU, the PL/LPG shows 1 row of the SKU with `Secondary Key` value set to 1, and the `Volume Discount` 0%.

The volume breakdown configurations are defined in the Product dimension, `<dependency>VolumeBreakdown PP` table. In order to use the feature properly, the table has to be configurable on the level of the lookup key and has one PP per dependency level. Example:


#### Price Parameter Values : GlobalVolumeBreakdown [3]

<input type="checkbox"/>	Business Unit	Product Gr...	Product Cl...	Volume 1	Discount 1	Volume 2	Discount 2	Volume 3	Discount 3
<input type="checkbox"/>	Food	Meatball	A	0	10.00 %	20	20.00 %	50	50.00 %
<input type="checkbox"/>	Others	*	*	10	10.00 %	20	20.00 %		

Exceptions on the SKU level are defined in the `VolumeBreakdownExceptions PP`. With this, the configuration in the corresponding `<dependency>VolumeBreakdown` will be ignored. Example:

#### Price Parameter Values : VolumeBreakdownExceptions [4]

<input type="checkbox"/>	SKU	Dependenc...	Volume #01	Discount #01	Volume #02	Discount #02	Volume #03	Discount #03	Volume #04
<input type="checkbox"/>	MB-0001	France	20	0.50 %	60	1.00 %			
<input type="checkbox"/>	MB-0001	Germany	20	0.50 %	60	1.00 %	100000	20.85 %	
<input type="checkbox"/>	MB-0002	Germany	5	5.00 %	10	10.00 %			

 For `<dependency>VolumeBreakdown` and `VolumeBreakdownExceptions PP` tables: If there is a space between the volume-discount pairs, then discount values cannot be parsed. Volume discount does not have an impact on Manual Override Price.

#### Margin Break-even and Revenue Break-even

When making pricing changes in the price list, there will be two added fields:

- Break-even volume for revenue
- Break-even volume for margin

This requires three fields:

- New Margin ( % )
- Previous Margin ( % )
- Previously approved price

The calculation formula:

- Break-even volume for revenue %:  $(\text{New Price} - \text{Previous Price}) / \text{Previous Price} * -100\%$
- Break-even volume for margin %:  $((\text{Old Price} - \text{Cost}) * \text{Volume}) / (\text{New Price} - \text{Cost}) / \text{Volume} - 1$

Example:

New				Old	Old	Breakeven	Old		New	Breakeven	New	New				
Old price	Price	Cost	Volume	Revenue	Margin	volume	margin %	margin %	Margin %	n margin	Margin	Revenue	breakeve	breakeve	New	New
						increase %				%		n volume	n volume	n volume	margin	Price
100	110	70	100	10000	3000	-9,0909091	0,3	0,36363636		-0,21212		75	90,90909		3000	10000
100	90	70	100	10000	3000	11,111111	0,3	0,22222222		0,35		150	111,1111		3000	10000

### Usage

To use this feature, set proper data and create a matrix PL/LPG and use the VolumeBreakdownMatrixLogic as the Matrix logic:

## Calculation Inputs

Allow distributed calculation

Allow column type change

Dynamic item mode :

Dynamic item filter : [Create Filter](#)

This calculation logic will be used if no specific method is defined in the product master data.

Default pricing logic :

**Matrix logic :**

This logic defines the secondary key set.

**Matrix logic element :**

Dynamic UOM :

Dynamic currency :

**Result Price :**

Auto-approve :

Manual Price Expiry :

Increase Threshold [%] :

Decrease Threshold [%] :

## Price Insights Dashboard

The Price Insights dashboard presents a summary of prices collected from a range of price lists and price grids. Results are presented in two columns per price list / price grid, showing Final Price and Final List Price for products. There are also three static lines showing historical data for the invoice price (minimum, maximum, average) based on existing transaction data.

It shows only prices calculated with logics made available through Accelerate Price Setting Package. Historical transaction data is also fetched using configuration of this package.

- [User Inputs](#)

- Analysis
- Data Requirements and Deployment

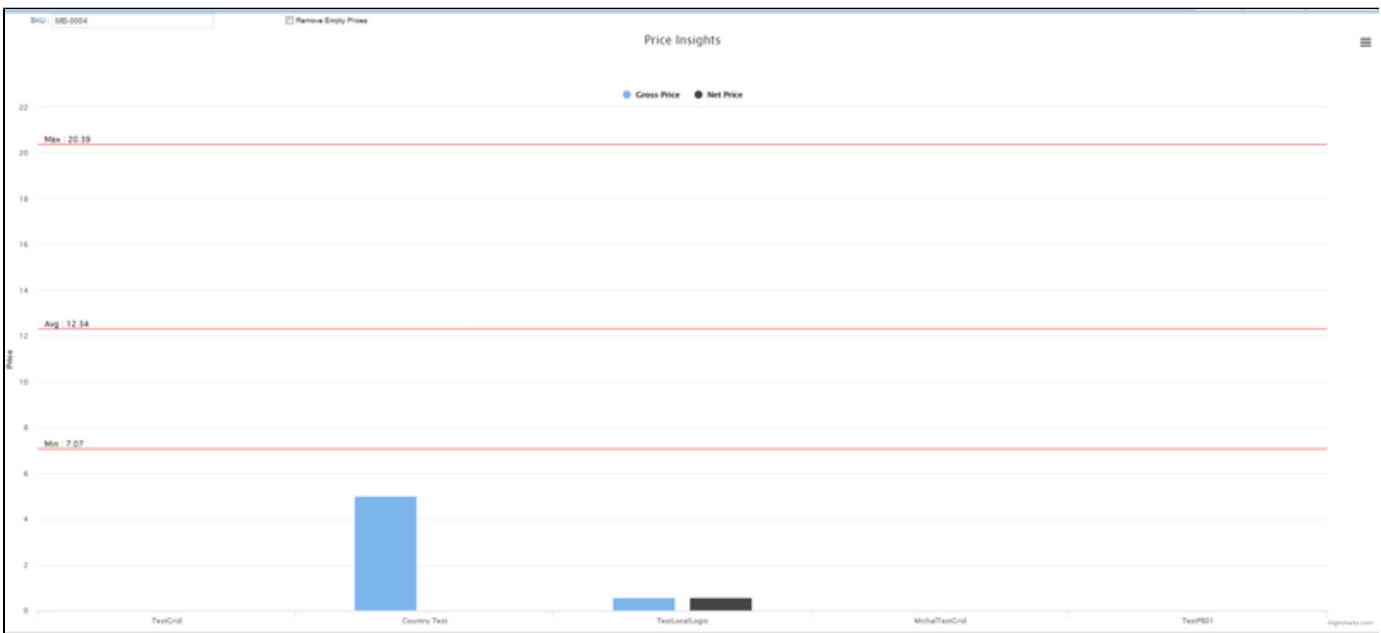
## User Inputs

SKU :   Remove Empty Prices Independent Level Name :

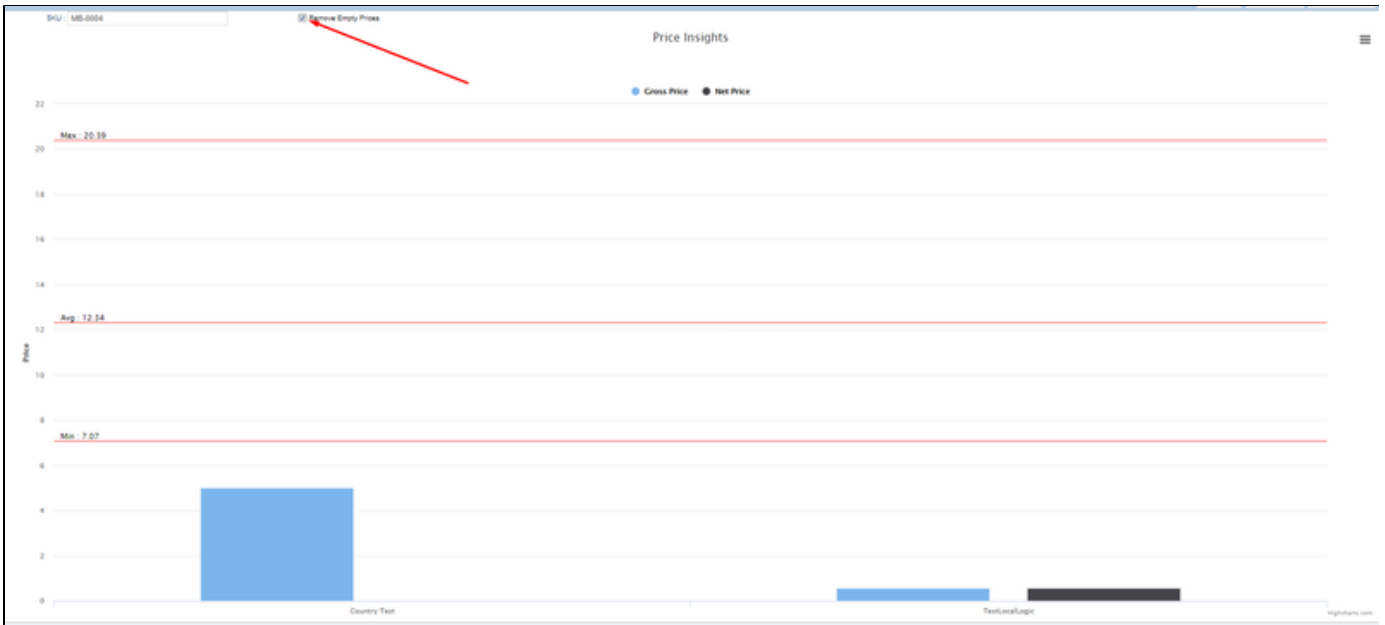
- **SKU** - SKU of the product you want to show in the dashboard.
- **Remove Empty Prices** - Defines if price lists and price grids with empty prices should be filtered out.
- **Independent Level Name** - Choose an independent level that you want to show. All price lists and price grids with the selected level name and those depending on this level will be included in the results.

## Analysis

After you configure the dashboard, you can get a result similar to this example:

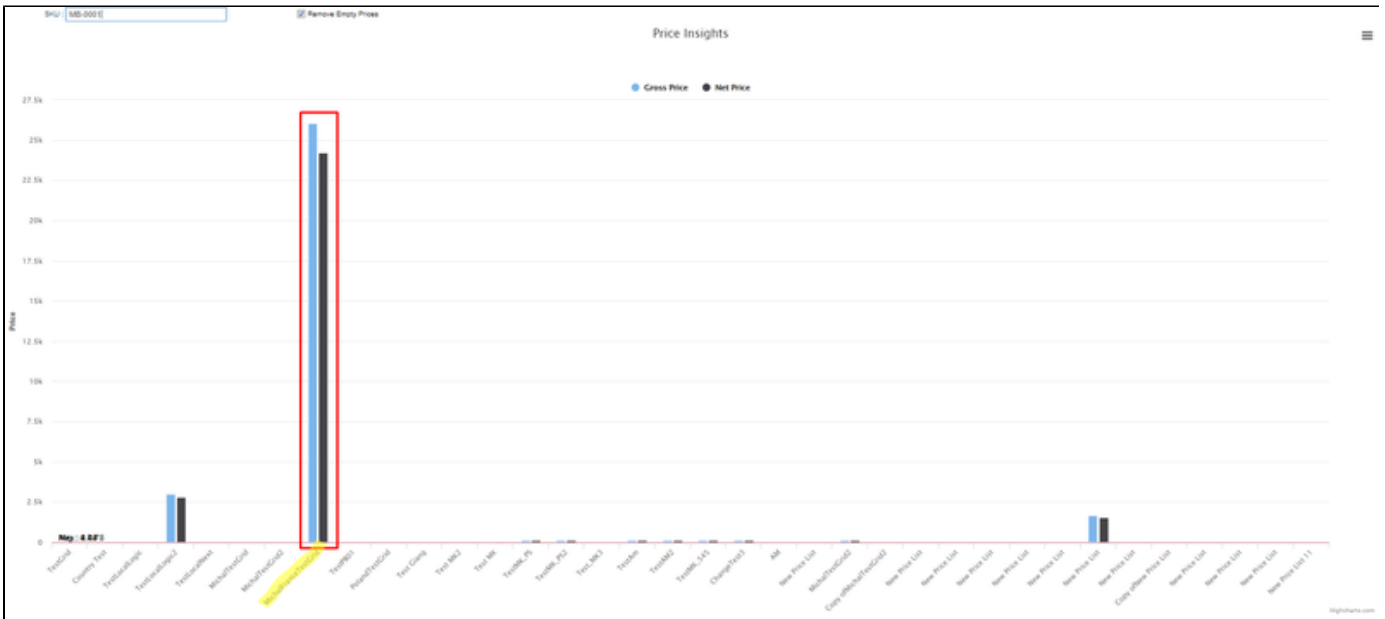


This dashboard presents data correlated with SKU MB-0004. There are three price lists / price grids with empty results, so this is something to be checked. You can filter out these empty results using the checkbox.



Now there are only calculated values. The actual prices are lower than the historical ones, so you can consider an increase.

This dashboard also helps you easily detect errors in calculations or exceptional behavior.



This part of the dashboard shows something irregular and so it is a good candidate for verification.

### Data Requirements and Deployment

The following parameters and data on a partition are needed for the dashboard to work.

Name	Description	Required
PL/PG	Any container that contains data presented in the dashboard	Yes
Data source for historical	Historical data about transactions	Yes

data		
Configuration in PP	Contains configuration for transaction Data Source	Yes

## Warning Handling

- [Warning Placement](#)
- [Warning Configuration](#)
- [Warning Codes List](#)

You can set up this Accelerator in many different ways. Almost every feature is configurable either by specifying where to look for necessary data or by telling the package how to handle this data. Because so many things can be adjusted, there is a lot of things that can go wrong as well. Missing configurations, missing data or conflicting business configurations are only a few examples.

To handle this we introduced a warning/error handling mechanism further described at <https://pricefx.atlassian.net/wiki/spaces/ACCDEV/pages/2608955463/Error+Handling>.

### Warning Placement

Warnings can be displayed at the following places:

- Default warning column
- Field where the issue happened (hidden if the element is internal)
- ResultMatrix at the end of the calculation

There are also some issues which prevent WarningManager from being initialized. Calculation will fail and an exception will be thrown in this case.

### Warning Configuration

Every warning code signals that something could not be executed properly; it is not always a business issue. There are cases like "EXCEPTION\_IGNORED" which just inform the user that a table override was ignored due to a manual override. It is possible to configure the visibility of each warning.

For every error, the following behavior can be configured:

- Raising any type of alert (only when the element is visible).
- Adding a default warning.
- Adding to a matrix element at the end of the calculation.

For details on setup see [WarningConfig PP](#).

The recommended approach on initial package deployment is to keep the default error setup until the package was fully configured and tested. Then all warnings related to unused features can be turned off by disabling alerts and hiding them from the default warnings popup. They can be also hidden from the custom Matrix popup, but we don't recommend it since it may make debugging future problems much harder.

### Warning Codes List

The table below represents a full list of expected errors and their default configuration that can be found in [WarningConfig PP](#) after package deployment:

Error Code	Message	Solution	Type	Alert	Matrix	Warning

CANT_APPLY_STRATEGY_CONDITIONS	The strategy condition(s) cannot be applied. Details are in technical message.	Verify the strategy conditions configuration table and its data.	Configuration	Red	Yes	Yes
CANT_CALCULATE_COST_TYPE	Unable to calculate cost for the specified type. Details are in technical message.	Verify the cost type configuration.	Configuration	Red	Yes	Yes
CANT_CALCULATE_MARGIN_BREAKEVEN_VOLUME	Unable to perform break-even calculation due to division by zero error.	Adjust the final price and cost.	Runtime	Red	Yes	Yes
CANT_CALCULATE_REVENUE_BREAKEVEN_VOLUME	Unable to perform break-even calculation due to division by zero error.	Adjust the final price.	Runtime	Red	Yes	Yes
CANT_GET_ACTUAL_PRICE_FROM_PL	Unable to get the actual price from price lists. No valid list found.	Make sure the price list with the same calculation logic as the current used has already existed and been approved.	Configuration		Yes	Yes
CANT_GET_CORRIDOR_CONFIG	Unable to get the corridor configuration.	Verify the corridor configuration tables setting.	Error	Red	Yes	Yes
CANT_GET_COST_TYPES_SELECTION	Unable to get the cost types selection configuration. Unable to calculate related element(s).	Verify cost types selection configuration table and its data.	Configuration	Critical	Yes	Yes

CANT_GET_DEPENDENCY_LEVEL_ADJUSTMENT	Unable to get the dependency adjustment value.	Verify dependency adjustment configuration table and its data.	Error	Red	Yes	Yes
CANT_GET_DISCOUNT	Unable to get the discount value.	Verify discount level configuration table.	Runtime	Red	Yes	Yes
CANT_GET_MIN_MARGIN	Unable to get the minimum margin.	Verify minimum margin configuration table and its data.	Error	Red	Yes	Yes
CANT_GET_PRODUCT_CHANGED_CAUSE	Unable to get the reason for change in the product.	Contact support.	Other	Red	Yes	Yes
CANT_GET_RELEVANT_COMPETITION_DATA	Unable to get the relevant competition data.	Verify relevant competition data configuration table and its data.	Error	Red	Yes	Yes
CANT_GET_STRATEGIES_SELECTION	Unable to get the strategies selection configuration. Unable to calculate related element(s).	Verify strategy selection configuration table and its data.	Error		Yes	Yes
CIRCULAR_DEPENDENCY	DependencyLevelConfiguration has circular dependency.		Runtime	Red	Yes	Yes
DEPENDENCY_LEVEL_ADJUSTMENT_IS_ZERO	There is no markup between Independent Price and this Pricing Level configured.	Check if you want to directly follow the Independent Level Price without any Markup.	Data		Yes	Yes
EMPTY_CONFIG	One of the mandatory fields in the configuration is empty.	Verify the configuration table and its data.	Fix config	Critical	Yes	Yes

	Details are in technical message.					
ERROR_LOOKING_UP_STRATEGY_CONDITIONS_DATA_PP	Unable to get Strategy Condition Price Parameter.	Verify configuration of strategy conditions.	Configuration	Red	Yes	Yes
ERROR_PARSING_COST_TYPE_DEFINITION	Unable to parse the definition of the specified cost type. Details are in technical message.	Verify the cost type configuration.	Configuration	Red	Yes	Yes
EXCEPTION_IGNORED	An exception value has been ignored.	Verify if the exception is valid and its value is valid.	Other	Red	Yes	Yes
EXCEPTION_STRATEGY_OVERRIDDEN	Strategy exception has been overridden by manual selection.		Business Warning	Yellow	Yes	Yes
FIELD_NOT_FOUND	The specified field is not found in the target table. Details are in technical message.	Verify the field name in the configuration.	Runtime	Red	Yes	Yes
INVALID_DIMENSION_LOOKUP_FIELD	The specified lookup keys in the current dimension are not found.	Verify the lookup keys of the current dimension in the pricing dimension configuration table.	AnyType	Red	Yes	Yes
INVALID_FINAL_PRICE	The calculated final price value is invalid.	Resolve critical and serious errors.	Error	Critical	Yes	Yes
INVALID_FORECAST_TYPE	The selected forecast type is not supported.	Verify current quarter Forecast setting.	Error	Red	Yes	Yes
			Data	Red	Yes	Yes

INVALID_INDEPENDENT_SOURCE_ID	The source ID of the specified independent price list to be referred is invalid.	Verify the Source ID in dependency configuration table. It should be the referred price list ID.				
INVALID_MIN_MARGIN_PERCENT	The minimum margin value is invalid.	Verify minimum margin configuration table data.	Data		Yes	No
CANT_GET_STOCK_CONFIG	Unable to get the stock configuration.	Verify Stock setting.	Runtime	Red	Yes	Yes
INVALID_VOLUME_BREAKDOWN_SETTING	Error in parsing volume breakdown due to invalid setting.	Verify volume breakdown configuration table and its data format, syntax.	Configuration	Red	Yes	Yes
MISSING_INPUTS_MARGIN_BREAKEVEN_VOLUME	Missing input (s) for margin breakeven volume. Required: cost, base price, volume, final price.	Verify the required parameter(s).	Runtime	Red	Yes	Yes
MISSING_INPUTS_REVENUE_BREAKEVEN_VOLUME	Missing input (s) for revenue break-even volume. Required: base price, volume, final price.	Verify the required parameter(s).	Runtime	Red	Yes	Yes
NO_ACTUAL_LIST_PRICE_FOUND	No valid actual list price found. Unable to calculate related element(s).	Verify Actual Price setting and its data.	Data	Yellow	Yes	No
NO_BASE_DEPENDENCY	There is no dependency for looking for non virtual dependency level.		Runtime	Red	Yes	Yes

NO_CONFIG	One of the mandatory configurations is not found. Details are in technical message.	Verify the configuration table and its data.	Fix config	Critical	Yes	Yes
NO_CORRIDOR_CONFIG_FOUND	No valid corridor configuration found.	Verify corridor configuration tables data.	Error		Yes	No
NO_COST_FOUND	No valid cost value found. Unable to calculate related element(s).	Verify Cost setting and its data.	Data		Yes	Yes
NO_DEPENDENCY_LEVEL_ADJUSTMENT	Unable to get data for dependency level adjustment.	Check dependency level adjustments data source.	Data	Red	Yes	Yes
NO_DISCOUNT_FOUND	No valid discount value found. Unable to calculate related element(s).	Verify discount level configuration table and its data.	Data	Red	Yes	Yes
NO_EXCHANGE_RATE_FOR_BATCHED_ITEM	One or more of the items in the batch has transaction data with not convertible currency.	Verify the exchange rate configuration and the data.	Runtime	Critical	Yes	Yes
NO_EXCHANGE_RATE_FOUND	No valid exchange rate value found.	Verify exchange rate configuration table and its data.	Data	Critical	Yes	Yes
NO_INDEPENDENT_LEVEL_CALCULATED_PRICES	No valid calculated prices from the independent level found.	Verify Prices of the independent item.	Error		Yes	Yes
NO_INDEPENDENT_FINAL_PRICE	No valid final price from the independent level found.	Verify Final Price of the independent item.	Error	Red	Yes	Yes

	Unable to calculate related element(s).					
NO_INDEPENDENT_LEVEL_PRICE	No valid price from the independent level found. Unable to calculate related element(s).	Verify Final Price of the independent item.	Error	Red	Yes	Yes
NO_INDEPENDENT_LEVEL_PRICE_DECISION	No valid price decision from the independent level found.	Verify Price Decision of the independent item.	Error	Red	Yes	Yes
NO_INDEPENDENT_LEVEL_RECORD_FOUND	No price record found for independent level.	Verify the corresponding price record at the independent level source.	Error	Yellow	Yes	Yes
NO_INPUT_FOR_CORRIDOR	Unable to calculate the price corridor due to missing parameter(s).	Verify the required parameter(s).	Error		Yes	No
NO_INPUT_FOR_DISCOUNT	Unable to get product discount data.	Check Discount Data.	Data		Yes	Yes
NO_INPUT_FOR_INDEPENDENT_LEVEL_ADJUSTED_PRICE	Unable to get Independent level adjusted price.	Check Independent Level Price and Dependency Level Adjustment.	Error		Yes	No
NO_INPUT_FOR_INDEPENDENT_LEVEL_PRICE_PRIORITY	Unable to get Independent Level Price Priority.	Check configuration.	Error		Yes	No
NO_INPUT_FOR_MARGIN	Unable to get margin.	Check value for margin in Price Parameter.	Data		Yes	Yes
NO_INPUT_FOR_MIN_MARGIN_PRICE	Unable to calculate the minimum margin price	Verify the required parameter(s).	Data		Yes	No

	due to missing parameter(s).					
NO_INPUT_FO R_MIN_MARGI N_VALIDATION	Unable to validate the minimum margin due to missing parameter(s).	Verify the required parameter(s).	Data	Red	Yes	Yes
NO_INPUT_FO R_NET_PRICE	Unable to calculate net price.	Check configuration of net price module.	Configuration		Yes	Yes
NO_INPUT_FO R_PRICE_CHA NGE_EFFECT	Unable to calculate the effect of price changing due to missing parameter(s).	Verify the required parameter(s).	AnyType	Red	Yes	Yes
NO_INPUT_FO R_STOCK_COV ER_DAYS	Unable to calculate stock cover days due to missing parameter(s).	Verify stock and sales volume forecast elements result data.	Runtime	Yellow	Yes	Yes
CANT_GET_V OLUME_DISCO UNT	Unable to get volume discount.	Check data for volume discount.	Data		Yes	Yes
NO_MIN_MAR GIN_CONFIG_F OUND	No valid minimum margin configuration found.	Verify minimum margin configuration table data.	Data		Yes	No
NO_MIN_MAR GIN_PRICE	Unable to calculate minimum margin price.	Check if everything is available for minimum margin price.	Data		Yes	No
NO_PF_TARG ET	Missing Price Flexibility Package target.	Contact support.	Configuration		No	No
NO_PRODUCT _CHANGED_C AUSE	Missing reason in Price Flexibility Package.	Contact support.	Other		Yes	Yes
NO_ROUNDIN G_RULE_FOU ND		Verify the rounding rules configuration	Configuration	Red	Yes	Yes

	There is no suitable rounding rule found.	table and its data.				
NO_SALES_VOLUME_FORECAST	There is no forecast for sales volume.	Verify the transaction data.	Data		Yes	No
NO_SALES_VOLUME_LAST_PERIOD	There is no sales volume in last period.	Verify the transaction data.	Data		Yes	No
NO_SALES_VOLUME_LAST_YEAR	There is no sales volume in last year.	Verify the transaction data.	Data		Yes	No
NO_SALES_VOLUME_YTD	There is no sales volume up-to-date.	Verify the transaction data.	Data		Yes	No
NO_STOCK_DATA	There is no valid record for stock.	Verify Stock setting and its data.	Runtime	Yellow	Yes	Yes
NO_STRATEGY_DEFINITION_FOUND	The strategy definition cannot be found.	Verify strategy definition configuration table and its data.	Configuration	Red	Yes	Yes
NO_SUITABLE_ROUNDING_RESULT	Applying the rounding rule makes the price invalid. The rule has been canceled.	Verify the price and the rounding rules configuration.	Configuration	Yellow	Yes	Yes
NO_TRANSACTION_SOURCE_TABLE_FOUND	The specified transaction table cannot be found.	Verify the Transaction Source setting.	Error	Red	Yes	Yes
NO_TURNOVER_FORECAST	There is no forecast for the turnover.	Verify the transaction data.	Data		Yes	No
NO_TURNOVER_LAST_PERIOD	There is no turnover in last period.	Verify the transaction data.	Data		Yes	No
NO_TURNOVER_LAST_YEAR	There is no turnover in last year.	Verify the transaction data.	Data		Yes	No
NO_TURNOVER_YTD	There is no turnover up-to-date.	Verify the transaction data.	Data		Yes	No

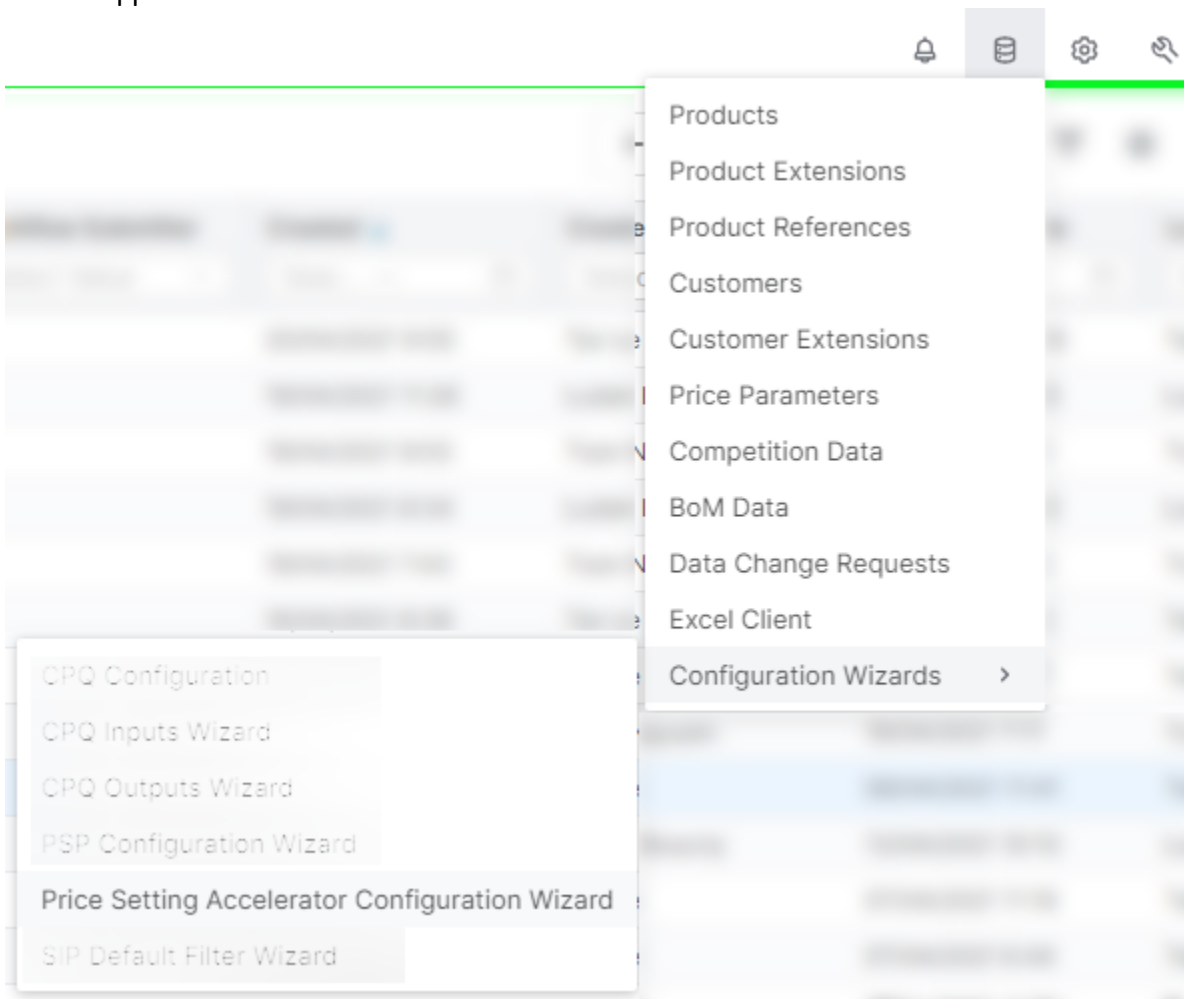
PP_VALIDITY_PERIODS_INVALID_FORMAT	Validity periods of the product are in invalid format.	Check Data and Configuration of PP. Validity period fields should have Date value type.	Runtime	Red	Yes	Yes
STRATEGY_OVERRIDE_PROHIBITED	Strategy selections are not available due to overridable setting.	If this is not the expected behavior, verify the Overridable property of member strategies definition.	Other		Yes	Yes
TOO_MANY_ROWS	There are too many records in the current batch.		Runtime	Yellow	Yes	Yes
TOO_MANY_ROWS_ABORTED	Unable to proceed because there are too many records in the current batch.		Runtime	Yellow	Yes	Yes
TOO_SMALL_MARGIN	The current margin is smaller than the specified minimum margin.	Verify the final price and the cost.	Business Warning	Yellow	Yes	Yes
UNABLE_TO_READ_TABLE_DATA	Data lookup could not be performed / results could not be read.		Runtime	Red	Yes	Yes
UNEXPECTED_ERROR	Unhandled error.	Contact support.	Error	Red	Yes	Yes
UNEXPECTED_PRICE_RANGE_FOR_CORRIDOR	Unexpected range for price corridor.	Check the configured corridor for price checks.	Data	Red	Yes	Yes
UNSUPPORTED_ACTUAL_PRICE_SOURCE_TYPE	The specified source type for actual price is unsupported.	Verify the source type of Actual Price setting.	Other	Red	Yes	Yes
			Configuration	Red	Yes	Yes

UNSUPPORTED_INDEPENDENT_SOURCE_TYPE	The source type of the specified independent price list to be referred is unsupported.	Verify the Source Type in dependency configuration table.				
VALIDITY_PERIODS_OVERLAPPED	Validity periods overlap.	Check validity period columns data.	Data	Red	Yes	Yes

## Price Setting Package Configuration Wizard

After deploying Price Setting Package you have to configure modules/features of the package. The package comes with a comprehensive wizard to help you with the configuration. With this wizard you can enable/disable and configure the different [Price Setting Modules](#).

The Price Setting Accelerator Configuration Wizard can be found in the default wizard section of the Pricefx application:



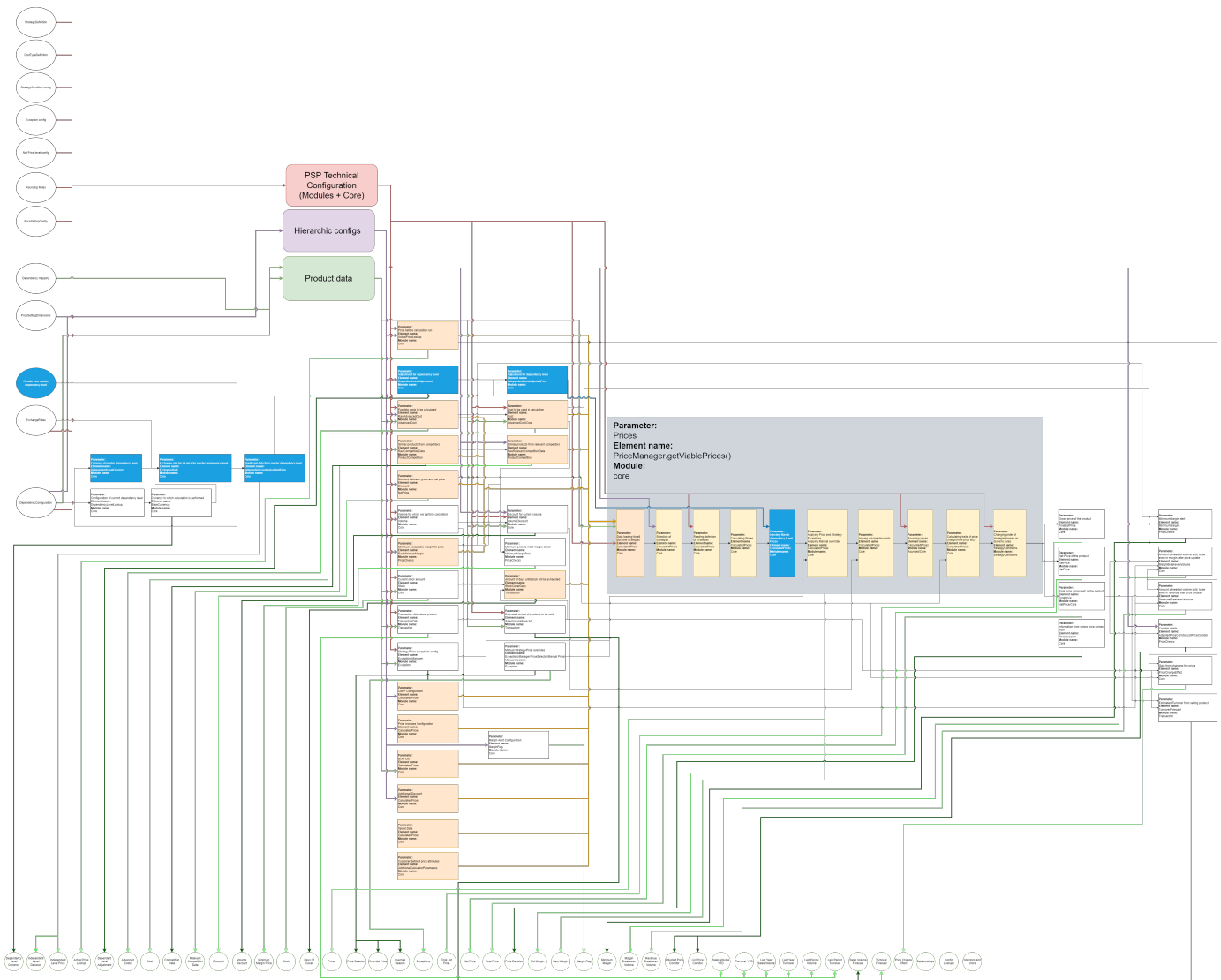
The Configuration Wizard is tuned to Pricefx Unity UI. It technically may work in Classic UI but the screen design will be broken. We strongly recommend to use it in Unity UI.

## Price Setting Fundamentals

The following diagram represents the data flow in the whole Price Setting Package (PSP). It looks intimidating, but it describes all the relations between different building pieces. It can help track where the results come from and how they are connected to each other.

Tips for reading the diagram:

- For simplification all configs are represented as 3 rectangles - Product Data, Hierarchic configs and PSP Technical Configuration.
- At the bottom, there are all elements visible to users from the PSP logic.
- Data only for Dependent Logic are shown in blue.
- Data used as input for strategy engines are shown in yellow.



## Dependency Mapping

- [Dependency Mapping Types](#)
- [Dependency Mapping Mechanism](#)
- [Dependent Price Lists and Data Fallbacks](#)

### Dependency Mapping Types

Dependency mapping defines how different lookup data will be filtered when loaded from a dependent price list. It is available in the [DependencyMappingConfig PP](#) and it directly impacts configuration found in [PriceSettingConfig PP](#).

#### Lookup

One source table for all dependencies. Example: use a PX product cost table for all continents:

SKU	Dependency	Cost
PD-0001	Global	10
PD-0001	Asia	11
PD-0001	EU	12
PD-0001	US	13
PD-0002	Global	14
PD-0003	Asia	15

For a PL/PG which has a dependency level Asia, the PD-0001 SKU will have a value of 11.

#### Table


Each source table for each dependency. Example: use multiple PX product cost tables for multiple continents:

Global table		US table	
SKU	Cost	SKU	Cost
PD-0001	10	PD-0001	11
PD-0002	11	PD-0002	12
PD-0003	12	PD-0003	13
Asia table		EU table	
SKU	Cost	SKU	Cost
PD-0001	12	PD-0001	13
PD-0002	13	PD-0002	14
PD-0003	14	PD-0003	15

For a PL/PG which has a dependency level Asia, the PD-0001 SKU will have a value of 12.

To use this type, change the name of the source table defined in PriceSettingConfig PP: the value should include the <<DependencyPreference>> placeholder. It will be replaced with the dependency property defined by the dependency mapping mechanism.

For example: for cost configuration which has dependency mapping type Table, the source table is Product Costs <<DependencyPreference>>, and the dependency value is DE, the dependency mapping mechanism will search for a table named Product Costs DE and then perform the lookup.

 The Dependency Field in DependencyMappingConfig PP is omitted in this type.  
Table dependency does not work for competitors because Pricefx has only 1 PCOMP table.


None

A source table for all dependencies. Example: use a PX product cost table for all:

SKU	Cost
PD-0001	10
PD-0001	11
PD-0001	12
PD-0001	13
PD-0002	14
PD-0003	15

For a PL/PG which has a dependency level Asia, the PD-0001 SKU will have a value of 10.

When a data element has the mapping type = None, it has no dependency mapping or hierarchy, or fallback. In case there are multiple records, it will fetch the first found. And when a table name is dependency wildcard but it has the mapping type = None, it uses the table with no wildcard. Example: <<DependencyPreference>>CostData => CostData.

 The Dependency Field and Mapping Source Field in DependencyMappingConfig PP are omitted in this type.

### Dependency Mapping Mechanism

Here you will find an example with Cost mapping.

#### Select Dependency

When creating a PL/PG with PSP, there is an input named Independent Level Name to select the dependency.

## Temp

### Calculation Inputs


Allow distributed calculation

Allow column type change

Dynamic item mode :

Dynamic item filter : [Create Filter](#)

This calculation logic will be used if no specific method is defined in the product master data.

Default pricing logic :  

Dynamic UOM :

Dynamic currency :

**Result Price :**

Auto-approve :

Manual Price Expiry :

Increase Threshold [%] :

Decrease Threshold [%] :

Cache lookup results

**Independent Level Name :**

The selected dependency is Global. It will then search in [DependencyConfiguration PP](#) for the corresponding record.

### Company Parameter Values : DependencyConfiguration [1]

<input type="checkbox"/>	global								
<input type="checkbox"/>	Dependency Level Na...	Depends ...	Source Type	Source...	Dimens...	Currency	Is com...	ISO C...	Sal...
<input type="checkbox"/>	Global	Independent	*	*	Area	USD	No	GG	SO00

With Lookup Mapping Type

In [DependencyMappingConfig PP](#), set the Dependency Field, Type, and Mapping Source Field values.

cost			
Name	Dependency Field	Type	Mapping Source Field
Cost	DependencyLevelName	Lookup	DependencyLevelName

- The `Type` is set to `Lookup` to indicate the mapping type.
- The `Dependency Field` points to `DependencyLevelName`. In the `Select The Dependency` step, the selected dependency is `Global`, and in `DependencyConfiguration PP`, the corresponding row has the `DependencyLevelName` field = `Global`. Therefore, the dependency value will be `Global`.
- The `Mapping Source Field` points to `DependencyLevelName`, it will search in the `Data Source` table for rows that have `DependencyLevelName` field value equaling to the dependency value (`Global` in this case).

The source table is defined in `PriceSettingConfig PP`.

Key	Condition	Type	Source	Source Table	Source Field	Source Field 2
Cost	*	Lookup	PX	CostData	Cost	Currency

It will search in the `CostData PX` table for records that have `DependencyLevelName` field equals to `Global` and get a value from the column pointed in the `Source Field` as `cost` value (`Cost` field).

With `Table Mapping Type`

In `DependencyMappingConfig PP`, set the `Dependency Field`, and `Type` values.

cost			
Name	Dependency Field	Type	Mapping Sourc...
Cost	DependencyLevel...	Table	

- The `Type` is set to `Table` to indicate the mapping type.
- The `Dependency Field` pointed to `DependencyLevelName`. In the `Select The Dependency` step, the selected dependency is `Global`, and in `DependencyConfiguration PP`, the corresponding row has the `DependencyLevelName` field = `Global`. Therefore, the dependency value will be `Global`.

The source table is defined in `PriceSettingConfig PP`.

Key	Condition	Type	Source	Source Table	Source Field	Source Field 2
Cost	*	Lookup	PX	<<DependencyPreference>>CostData...	Cost	Currency

It will search in the `GlobalCostData PX` table for records and get a value from the column pointed in the `Source Field` as `cost` value (`Cost` field).

With `None Mapping Type`

In `DependencyMappingConfig PP`, set the `Type` value.

cost			
Name	Dependency Field	Type	Mapping Sourc...
Cost		None	

- The `Type` is set to `None` to indicate the mapping type.

The source table is defined in `PriceSettingConfig PP`.

Key	Condition	Type	Source	Source Table	Source Field	Source Field 2
Cost	*	Lookup	PX	CostData	Cost	Currency

It will search in the `CostData PX` table and get a value from the column pointed in the `Source Field` as cost value (`Cost` field).

## Dependent Price Lists and Data Fallbacks

The Price Setting Package supports multiple price lists / price grids. Each of the lists is connected with one of the dependency levels: [DependencyConfiguration PP](#). This is set on the PL/PG level as an input parameter, after choosing the proper logic. This structure is responsible for the following features:

- **Dependency mapping** - Allows having more than one data entry for each product, depending on the calculation context (e.g. different costs for web shop and brick and mortar shop),
- **Data fallback mechanism** - Allows incomplete data in case of granular pricing. This way only unique data needs to be put to the most granular dependency levels; the rest will fallback to the master dependencies.
- **Lookup keys config fallback mechanism** - Allows incomplete configurations for dimensional configurations. The mechanism is similar to the data fallback mechanism, with some more intuitive tweaks to configurations.
- **Master price adjustments** - Allows centralized approach to pricing where prices from the master price list are important for the dependent logic.
- **Grouping transaction data** - Allows the master price list to be completely valid. Transaction data of the dependent price lists will usually be (except for the HQ mode described below) also part of the master dependency.

In this section:

- [Independent Logic](#)
- [Dependent Logic](#)
  - [Dependency Mapping](#)
  - [HQ Dependency Mode](#)
  - [Price Adjustments](#)
  - [Data Lookup Fallbacks](#)
  - [Lookup Keys Config Fallbacks](#)
  - [Transaction Data Fallback](#)

### Independent Logic

To set up independent price lists / price grids, the "IndependentPriceListLogic" logic must be used to run calculations. As the name suggests, these PGs/PLs are independent objects and all the calculation results depend only on the package configurations. This mode can be used to handle the simplest scenarios where only one dependency level exists or as a root for more complex independent-dependent hierarchies.

## Dependent Logic

To set up dependent price lists / price grids, the "DependentPriceListLogic" logic must be used to run calculations. During PG/PL configuration you will be asked to select both current dependency level and parent dependency level. This connection will be used throughout the whole calculation to base prices on parent prices or to utilize hierarchical data fallbacks.

### Dependency Mapping

To understand how to create distinct entries of the same product data which are unique in the context of a defined dependency level, see [DependencyMappingConfig PP](#).


### HQ Dependency Mode

If two dependencies have one dimension and one depends on another, then we call such relationship an HQ relationship. HQ dependency levels will be skipped when looking for fallbacks but the master price might be looked up from such dependency.

### Price Adjustments

Dependent price lists / price grids can use their master result prices as base for their own prices. Independent price is always taken from the first found master in the hierarchy but there are two exceptions:

- Independent logic does not expect a master price.
- If there is no master price list configured for the dependency level, then the master dependency is skipped. Such master dependency is called "virtual" and it exists only for fallback purposes.

 Currently, it is not possible to create a virtual dependency level on top of the hierarchy, as anything below the top uses the dependent logic - and expects to have a master price somewhere in the tree.

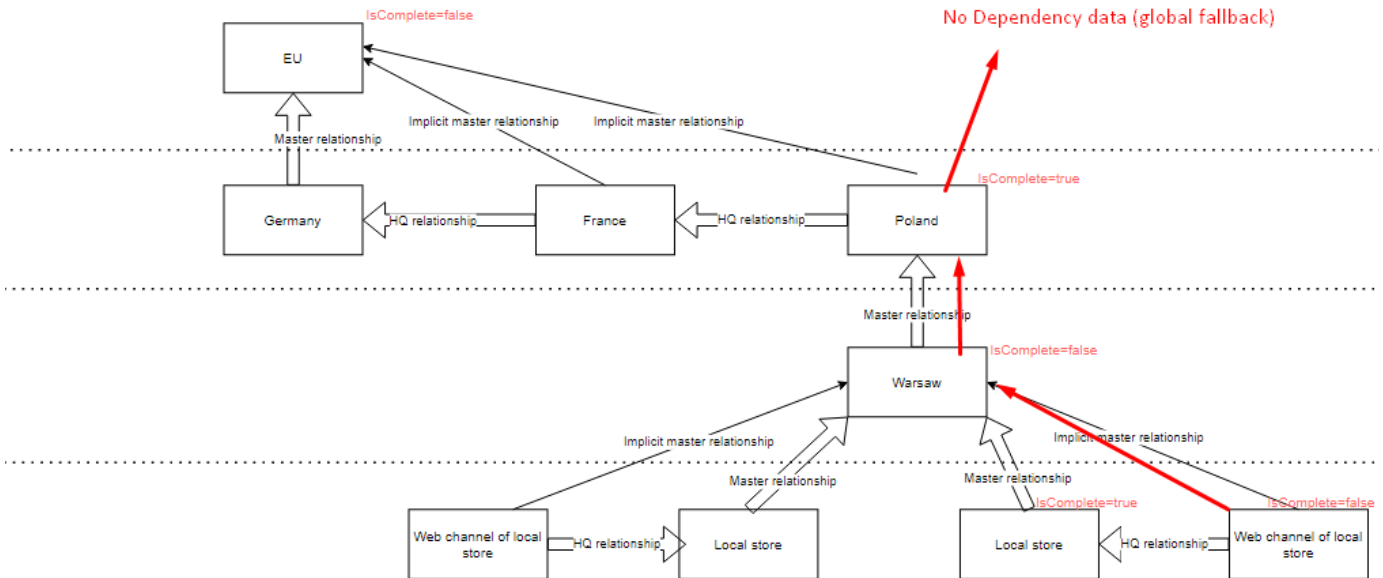
### Data Lookup Fallbacks

Fallbacks follow the following rules:

- Check if the current dependency level is complete.
  - If yes, abort the algorithm.
- Look up the master dependency level.
  - If the master dependency level does not exist, add empty mapping data as a possible fallback and abort the algorithm.
  - If the master dependency level is not in the HQ relationship with the current level, then add a master as a possible fallback.
- Repeat with the master dependency level as the current dependency level.

Example without the isComplete flags:





For a given configuration, when looking up the CostPlus config for a web channel of a local store, the algorithm will look for configs in PPs in this order:

- WebchanneloflocalstoreCostPlus
- WarsawCostPlus
- PolandCostPlus
- CostPlus

#### Transaction Data Fallback

Transaction data from <https://pricefx.atlassian.net/wiki/pages/resumedraft.action?draftId=3246261385> is a special case. More information can be found [here](#).

#### Data Lookups

PSP uses LookupManager for data lookups. Most of them are “standardized”, they are called in the same way. Some of them differ, due to different sources or data structure.

Flow of standardized lookups:

1. Once for a batch (by default 200 products) DB lookup is performed.
  - a. User entered source type is converted to the proper database type (e.g. PP to MLTV4).
  - b. The first key of the database type is read; in standardized lookups it is always SKU.
  - c. Validator closures are loaded. These will be used later.
  - d. SortBy parameter is prepared. It is there only for technical purposes (finding data with proper validAfter date), otherwise no sorting is performed.
  - e. Filters are constructed.
    - i. ValidityDates (if used) are in proper ranges. PSP supports “ValidAfter” mode and “ValidAfter-ValidTo” mode.
    - ii. Data load is prepared for any SKU in the batch.

- iii. Data is placed in a proper table. It might be a simple filter, or it might be a complex one when using Dependency Mapping with the "Table" type (the whole hierarchy is looked up).
  - iv. Data is prepared for proper dependency levels (the whole hierarchy). Used only when the Dependency Mapping with "Lookup" type is used.
  - f. Data is looked up.
  - g. Data is grouped per SKU (if lookup did not fail).
  - h. For every SKU the group validators are run (if lookup did not fail).
    - i. Validators on the data entry level - each Dependency Level in the hierarchy has its own closure. It marks every entry if it belongs to the given dependency level. It is especially important in the "Table" Dependency Mapping where entries from multiple tables are mixed together.
    - ii. Validator on the entry set level - marks the whole set of entries and informs if data overlap. It also considers the fallback mechanism. If there is any data for the first dependency level, it checks only these entries.
  - i. Data lookup is registered for debugging (registering keeps data for other products in the batch).
2. It is checked if data lookup threw any exception. Any exceptions are propagated further for every product in the batch, so that config issues are easier to debug.
  3. Data for SKU is read.
  4. Warning is registered if data has overlapped (only in "ValidFrom-ValidTo" mode).
  5. Proper data for product is returned.

Further in this section:

- [Standardized Lookups](#)
- [Special Cases Lookups](#)
- [Hierarchical Lookups](#)

## Standardized Lookups

PSP uses LookupManager for data lookups. Most of them are "standardized", they are called in the same way. Some of them differ, due to different sources or data structures.

The flow of standardized lookups:

1. Once for a batch (by default 200 products), a DB lookup is performed.
  - a. User entered source type is converted to the proper database type (e.g. PP to MLTV4).
  - b. The first key of the database type is read; in standardized lookups, it is always SKU.
  - c. Validator closures are loaded. These will be used later.
  - d. SortBy parameter is prepared. It is there only for technical purposes (finding data with proper valid after date), otherwise, no sorting is performed.
  - e. Filters are constructed.
    - i. ValidityDates (if used) are in proper ranges. PSP supports "ValidAfter" mode and "ValidAfter-ValidTo" mode.
    - ii. Data load is prepared for any SKU in the batch.
    - iii. Data is placed in a proper table. It might be a simple filter, or it might be a complex one when using Dependency Mapping with the "Table" type (the whole hierarchy is looked up).
    - iv. Data is prepared for proper dependency levels (the whole hierarchy). It is used only when the Dependency Mapping with "Lookup" type is used.
  - f. Data is looked up.
  - g. Data is grouped per SKU (if lookup did not fail).
  - h. For every SKU the group validators are run (if lookup did not fail).

- i. Validators on the data entry level - each Dependency Level in the hierarchy has its own closure. It marks every entry if it belongs to the given dependency level. It is especially important in the "Table" Dependency Mapping where entries from multiple tables are mixed together.
  - ii. Validator on the entry set level - marks the whole set of entries and informs if data overlap. It also considers the fallback mechanism. If there is any data for the first dependency level, it checks only these entries.
    - i. Data lookup is registered for debugging (registering keeps data for other products in the batch).
2. It is checked if data lookup threw any exception. Any exceptions are propagated further for every product in the batch so that config issues are easier to debug.
  3. Data for SKU is read.
  4. A warning is registered if data has overlapped (only in the "ValidFrom-ValidTo" mode).
  5. Proper data for the product is returned.

## Special Cases Lookups

### Actual Price

Technically, the Actual Price PX lookup is standardized. However, from the configuration point of view, the flow of data reading is non-standard when using PL and PG sources.

PL and PG do not use batching. PG is not even a lookup as such, since it reads data from the previous run of the same price grid.

Preference: [Actual Price Lookup](#)

### Transaction

- Hierarchy works differently for Transaction data (it goes down for all sub-levels of the dependency, instead of working as a fallback).
- A lot of data will be returned for the given time period, there is a "date overlap" issue.
- Lookup manager supports `api.stream` calls, but transaction data might be stored in a Datamart or Data Source.

Preference: [Transaction Lookup](#)

### Forecast

- Hierarchy works differently for Transaction data (it goes down for all sub-levels of dependency, instead of working as fallback).
- A lot of data will be returned for a given time period, there is a "date overlap" issue.
- Lookup manager supports `api.stream` calls, while transaction data might be stored in Datamart or Data Source.

Preference: [Forecast Lookup](#)

### Rounding Rules

We do not perform lookup per SKU.

Preference: [Rounding Rules Lookup](#)

### Competition Data

- No user-config ("ValidFrom", "ValidFrom-ValidTo" modes are not supported).

- Only PCOMP as a source (single table, no table name configuration).
- No Dependency Mapping with the "Table" type supported (single table, cannot create table per dependency level).

## Hierarchical Lookups

- [Configuration Of Bootstrapping](#)
- [Hierarchical Config Lookup](#)
- [Lookup List](#)

## Configuration Of Bootstrapping

Configuration tables for Hierarchical Lookups are created dynamically by bootstrapping. Bootstrapping receives 4 inputs:

- There is 1-5 hierarchical attributes configured for any lookup in [PriceSettingDimensions PP](#). These are the keys of newly created Price Parameters. Hierarchical lookups are configs not intended to be configured for a product, but for a group of products. However, the decision how to split products into groups is up to the user. Splitting products per SKU into 1-element group will work just fine.
- There are 13 features to be configured and one general fallback. It decides which configs (from above) are put in which Price Parameters as keys.
- Most of the Hierarchical Lookups have one table per Dependency Level + 1 (universal fallback). Dependency Configuration should be prepared before bootstrapping: [DependencyConfiguration PP](#)
- Bootstrapping expects to find on the partition the below listed Price Parameters. These PPs will be removed during the run. Since the amount of PPs might be very big (for a lot of Dependency Levels), each Price Parameter is placed where related PPs should be generated.
  - AdditionalDiscountTempHook
  - AdjustedPriceCorridorTempHook
  - BaseStrategySelectionTempHook
  - CostPlusTempHook
  - CostSelectionTempHook
  - DependencyLevelAdjustmentTempHook
  - DiscountTempHook
  - ListPriceCorridorTempHook
  - MinMarginTempHook
  - PriceIncreaseTempHook
  - RelevantCompetitionDataTempHook
  - StrategySelectionTempHook
  - VolumeBreakdownTempHook

**i** All of these are handled by PlatformManager in the standard deployment scenario. For edits after deployment, follow [Changing product segmentation](#).

## Hierarchical Config Lookup

After generating hierarchical tables, these should be filled with data:

### Attributes

Attributes of hierarchical tables are described on their pages.

### Keys

Each hierarchical table has 1-5 keys. Each key is a product attribute. If there was no name to the product column, "ProductColumn-attributeXX" name is used. The user should describe groups of products, with the ability of using "\*" fallback. Order of entries is irrelevant - the most detailed config is chosen.

If no entry has been chosen, the user might create a general fallback with only an asterisk ("\*").

If no "asterisk fallback" is used, hierarchical fallback will be utilized. It means there is no need to create configs for very detailed dependency levels on which we do not perform segmentation: <https://pricefx.atlassian.net/wiki/spaces/ACCDEV/pages/2962817818/Dependent+Price+Lists+and+Data+Fallbacks#Lookup-Keys-Config-Fallbacks>.

Example:

<input type="checkbox"/>	Business Unit	ProductColumn-attribute10	Product Class
<input type="checkbox"/>	Food	*	*
<input type="checkbox"/>	Food	Meatball	C
<input type="checkbox"/>	Others	*	*
<input type="checkbox"/>	*	*	*
<input type="checkbox"/>	Beverages	*	*
<input type="checkbox"/>	Food	Meatball	B
<input type="checkbox"/>	Others	Toppings	C
<input type="checkbox"/>	Food	Apple	A
<input type="checkbox"/>	Food	Meatball	tempA
<input type="checkbox"/>	Food	Sausage	A
<input type="checkbox"/>	Food	Sausage	B
<input type="checkbox"/>	Test PLCM	*	*
<input type="checkbox"/>	Food	Apple	B
<input type="checkbox"/>	Food	Meatball	D
<input type="checkbox"/>	Food	Meatball	A
<input type="checkbox"/>	Beverages	Non-Alcoholic	C

## Lookup List

### Price Calculation

- [Calculation Engines](#)
- [Price Strategies](#)
- [Strategy Importance](#)

### Calculation Engines

Calculation Engines provide plug-in/plug-out methods for price calculation which can be used in Price Setting Package and other projects as a standalone library.

Any given engine can be passed a simple Price Parameter (type MATRIX) with keys and values as specified in the Additional Configuration column in detailed documentation of the engine.

All available strategies are configured using the StrategyDefinition Price Parameter. More information about it can be found in [Other Configs](#) and in documentation for each engine.

Documentation for all engines is available as part of the PSP documentation. More details can also be found in the Groovy documentation of the engines, available in this repository: [CalculationEnginesLib](#).

Here is a short explanation of how the engines work and what they can be used for:

Engine Name (click for details)	Functionality	Sample Supported Strategies
<a href="#">Adjustment Engine</a>	Takes one price as a base and applies a factor to it.	<ul style="list-style-type: none"> <li>• Cost Plus</li> <li>• Price Increase</li> </ul>
<a href="#">Anchor Engine</a>	Calculates prices based on the price of another SKU.  Note: This engine is deprecated.	<ul style="list-style-type: none"> <li>• Anchor Pricing</li> </ul>
<a href="https://pricefx.atlassian.net/wiki/spaces/ACCDEV/pages/2517468375/Attribute+Engine">https://pricefx.atlassian.net/wiki/spaces/ACCDEV/pages/2517468375/Attribute+Engine</a>	Calculates prices based on the price of another SKU and current SKU's attributes impact value.	<ul style="list-style-type: none"> <li>• Anchor Pricing</li> </ul>
<a href="#">Competition Engine</a>	Calculates prices according to existing competition prices.	<ul style="list-style-type: none"> <li>• Competition Based Pricing</li> </ul>
<a href="#">Kit Engine</a>	Calculates prices of a kit based on subcomponent prices.	<ul style="list-style-type: none"> <li>• Kit Pricing</li> </ul>
<a href="#">Lookup Engine</a>	Looks up prices from an existing table.	<ul style="list-style-type: none"> <li>• RRP</li> <li>• Promotion Pricing</li> <li>• Everything that works with a lookup of a stored price</li> </ul>
<a href="#">Net Engine</a>	Calculates a gross price of a product, based on a specific "pocket price" and discounts. The pocket price is always looked up using the LookupEngine.	<ul style="list-style-type: none"> <li>• Net Pricing</li> </ul>
<a href="#">Custom Engines</a>	Any custom library method can be used as an engine as long as it takes proper parameters and returns a proper result.	<ul style="list-style-type: none"> <li>• Basically anything</li> </ul>

Currently, with our standard out-of-the-box configuration, the package comes preconfigured with following strategies:

- Minimum Competition Based Price
- Average Competition Based Price
- Maximum Competition Based Price
- Recommended Retail Price
- Cost+
- Price Increase
- Kit Pricing
- Anchor Pricing

## Adjustment Engine

Adjustment Engine takes care of simple “Value + Adjustment” calculations. It is used to implement strategies such as Cost+ or Price Increase.

### Input Parameters

Input	Type	Description
Value	Big Decimal	Used as a base for adjustments.
Adjustment	Big Decimal	Used as an adjustment. Can be absolute value or percentage - it depends on the mode selected in Additional Engine Configuration. For percentage based calculations the expected range is 0.0-1.0.

### Additional Engine Configuration

This engine needs an additional simple configuration Price Parameter where all Configuration Options are present as keys.

Configuration Option	Expected value	Description
Calculation Mode	“Absolute”	Result is: $\text{Value} + \text{Adjustment}$
	“Percentage”	Adjustment is a percentage. Result is: $\text{Value} * (1 + \text{Adjustment})$
	“SellingPrice”	Adjustment is a percentage. Result is: $\text{Value} / (1 - \text{Adjustment})$

### Default Strategy Calculation Parameters

For Cost+: `PRODUCT_COST, PLUS_FOR_PRODUCT`

For Price Increase: `BASE_PRICE, PRICE_INCREASE`

## Attribute Based Engine

The idea of attribute-based pricing is to define the price based on product attributes like color, weight etc. It is based on the Anchor Follower approach: take the price of another SKU, then modify it to get the final price.

The engine supports only one level which means there can only be pairs like SKU A SKU B. A chain of anchors like SKU A SKU B SKU C ... is not supported.

**Note:** If the used Price List or Live Price Grid is of the Matrix type, the engine assumes that the second key is used for storing the volume information and will search for an anchor product with the secondary key equaling to 1.

In this section:

- [Understanding the Calculation Mechanism](#)
  - [Warnings](#)
- [How to Use It](#)
  - [Input Parameters](#)
  - [Additional Engine Configuration](#)
  - [Default Strategy Calculation Parameters](#)
  - [Define Attribute Data \(with Sample Data\)](#)

### Understanding the Calculation Mechanism

The formula is defined in the AttributeBasedPricingRules PP. It takes the price of the anchor SKU as the basis, then calculates from left to right; there are no additional math operations. For each attribute in the rule:

- It gets the current product attribute value. The way to find it is defined in the PricingAttributes PP.
- It gets the impact of the above value. The impact value can be Value-Based or Interval-Based. It is defined in the ValueAttributesConversion PP or IntervalAttributesConversion PP accordingly.
- It calculates with the impact value.

### Warnings

- A rule is considered to be invalid if it is not a continuous string or it is a continuous string but ends with an operator. Example:

<input type="checkbox"/> Valid	+	AttributeP	+	AttributePX	+	AttributePP
<input type="checkbox"/> Invalid	+	AttributeP	+	AttributePX		AttributePP
<input type="checkbox"/> Invalid1	+	AttributeP	+	AttributePX	+	AttributePP +

- Divide by zero is not allowed.
- If it is configured to have validity periods and the data overlap, an exception is thrown.
- If it is configured to have dependency mapping but no data match the criteria, it gets the first one with the null value in the filter field.
- Any invalid field name / rule name / attribute name / ... is not allowed.
- This engine uses the "Dirty Run" functionality of Pricefx. You should not mix products that are used by other engines that mark items as dirty (e.g. Kit Engine) in one PL/LPG. It can lead to undefined behavior and wrong results.
- This engine only works correctly when all related products are in the current calculated Price List or Live Price Grid.

### How to Use It

#### Input Parameters

Input	Type	Description
SKU	String	SKU of the calculated product.
Final List Price Element Name	String	Name of the element that keeps the Final List Price. It will be used for the Anchor price lookup if the calculation is in the Net mode.
Final Price Element Name	String	Name of the element that has the Final Price. It will be used for

		Anchor price lookup if the Final List Price element is empty. It usually happens during the Gross calculation.
Dependency Properties	String	Properties of current dependency.
Strategy Name	String	Name of the strategy. It is used to connect the name of the attribute based strategy (in <i>PP AttributeBasedPricingRules</i> ).  You can use the constant <code>STRATEGY_NAME</code> to connect it with the name of the strategy.

*Additional Engine Configuration*

This engine needs an additional simple configuration Price Parameter where all Configuration Options are present as keys.

Configuration Option	Expected Value	Description
Source Type	Allowed values: <ul style="list-style-type: none"> <li>• PX</li> <li>• PP</li> <li>• P</li> <li>• PXREF</li> </ul>	Defines where the anchor data is kept.
Source Table	ExampleTableName	Name of the data table. Expected only when PX or PP <i>Source Type</i> is used.
Anchor Label	ExampleAnchorColumn	Name of the column that contains SKU of the anchor product.
Sku Field Label	ExampleSkuColumn	Name of the column that contains SKU of a currently calculated product.

*Default Strategy Calculation Parameters*

SKU, FINAL\_LIST\_PRICE\_ELEMENT\_NAME, FINAL\_PRICE\_ELEMENT\_NAME, DEPENDENCY\_PROPERTIES, STRATEGY\_NAME

*Define Attribute Data (with Sample Data)*

AttributeBasedPricingRules PP

Name	Operator #1	Pricing Attribute #1	Operator #...	Pricing Attribute #...
	+	Color	+	Size

Attribute-Based Simple				
...	...	...	/	0

- Fields:
  - Name - Rule name
  - Operator #XX - Supported operators: +, -, \*, /
  - Attribute #XX - Existing pricing attribute name in PricingAttributes PP

### PricingAttributes PP

Pricing Attribute (Key)	Type	Source Type	Source Name	Source Field	Dependency Field	Dependency Mapping Type	Mapping Source Field	ValidFrom Field Name	ValidTo Field Name
Color	Value	P		Color					
Size	Interval	PX	Additional Product Data	Size	Country	Lookup	Country	ValidFrom	ValidTo
Weight	Direct Value	P		Weight					

- Fields:
  - Pricing Attribute - Name of the attribute
  - Type - Attribute value type
    - Value - Single value
    - Interval - Value in a specified range
    - Direct Value - Impact value is also the attribute value, no conversion for this type. The value data type can only be a number.
    - Direct Value and Interval Type only works for numeric values.
  - Source Type - Source table type
    - P / PX / PP
  - Source Name - Name of the source table
  - Source Field - Name of the field in the source table to take attribute value
  - Dependency Field - Name of the field in the dependency configuration table to take dependency value
  - Dependency Mapping Type - Type of dependency mapping
    - Lookup / Table
    - Fallback on dependency mapping - Returns one with "null" in the dependency field when there is no specific one
  - Mapping Source Field - Name of the field in the source table to take the matching value
  - Valid From Field Name - Name of the field in the source table to take the beginning date of the validity period
  - Valid To Field Name - Name of the field in the source table to take the end date of the validity period

### ValueAttributesConversion PP

Pricing Attribute	Pricing Attribute Value	Price Impact Value
Color	red	3
Color	blue	2

Color	2	1
Color	<<fallback>>	1.5


Mapping one attribute value to one impact value. The attribute value can be both string and number.

#### IntervalAttributesConversion PP

Pricing Attribute	Pricing Attribute Value From	Pricing Attribute Value To (including)	Price Impact Value
Size	0	10	1.2
Size	10	50	1.4
Size	50	999999999	1.5

Mapping many attributes value to one impact value. The attribute value can only be number.

#### Anchor Engine

 This engine is deprecated. Use the [Attribute-based engine](#) instead.

Anchor Engine calculates a price for a given product based on another product's price (Anchor) and anchor factor by which we multiply this price. The formula is:  $Price = AnchorPrice * (1 + AnchorFactor)$

This engine supports only one level of connection. It means you cannot specify an anchor for an anchor, etc. In addition - this engine works properly only when all connected products are added to the same Price List or Live Price Grid.

#### Important notes:

- This engine uses the "Dirty Run" functionality of Pricefx. It means that you should not mix products that are used by other engines that mark items as dirty (e.g. Kit Engine) in one PL/LPG. It can lead to undefined behavior and wrong results.
- Because this engine doesn't return a price after the first calculation, if it is used as Base Strategy, it will not be shown in the "Prices" popup. The strategy will appear only when the second calculation run completes.
- If the used Price List or Live Price Grid are of the Matrix type, the engine assumes that the secondary key is used for storing the volume information and will search for an anchor product with the secondary key equaling to 1.

#### Input Parameters

Input	Type	Description
SKU	String	SKU of the calculated product.
Final List Price Element Name	String	Name of the element that keeps the Final List Price. It will be used for the Anchor price lookup if the calculation is in the Net mode.
Final Price Element Name	String	

		Name of the element that has the Final Price. It will be used for Anchor price lookup if the Final List Price element is empty. It usually happens during Gross calculation.
--	--	--

Additional Engine Configuration

This engine needs an additional simple configuration Price Parameter where all Configuration Options are present as keys.

Configuration Option	Expected value	Description
Source Type	Allowed values: <ul style="list-style-type: none"> <li>• PX</li> <li>• PP</li> <li>• P</li> <li>• PXREF</li> </ul>	Defines where the anchor data is kept.
Source Table	ExampleTableName	Name of the data table. Expected only when PX or PP <i>Source Type</i> is used.
Anchor Label	ExampleAnchorColumn	Name of the column that contains SKU of the anchor product.
Factor to Anchor Field Label	ExampleFactorColumn	Name of the column that contains the value used as factor multiplication.
Sku Field Label	ExampleSkuColumn	Name of the column that contains SKU of a currently calculated product.

Default Strategy Calculation Parameters

SKU , FINAL\_LIST\_PRICE\_ELEMENT\_NAME , FINAL\_PRICE\_ELEMENT\_NAME

**Competition Engine**

Competition Engine supports various options for price calculation based on competitor prices (defined through the engine’s Additional Configuration PP table).

Competitor price can be selected based on one of two approaches:

- Competitor Position - Select one competitor to align the price with.
  - Min/max - Select a minimum/maximum available competitor price.
  - min + X / max - Y - Select minimum/maximum competitor price position and adjust it by the given value.

- 10%, 50%, 70% - Select the target competitor based on the provided percentage. The formula for the calculation is:  $\text{TargetCompetitorPosition} = \text{NumberOfCompetitors} * \text{Percentage}$
- Price Position - Select a price at the given percentage point. A value of 0% matches the lowest competition price and value of 100% matches the highest competition price. The formula for calculation is:  $\text{Price} = \text{CompetitorMinPrice} + (\text{CompetitorMaxPrice} - \text{CompetitorMinPrice}) * \text{Percentage}$

**Note:** Competitor Position and Price Position cannot be used at the same time.

After the competitor price has been selected, the engine will find the corresponding competitor name if in the "Competitor Position" mode.

Then you can additionally "reposition" the price by:

- Percentage - Modifies the price by a provided percentage. To make the price 5% cheaper, you use -5 ; the same applies for a positive adjustment.
- Absolute value - Modifies the price by an absolute value. To make the price 10 units cheaper, you use -10; the same applies for a positive adjustment.

The engine supports Force Margin Check to verify that the selected competitor price is affordable. You can set values "Yes/No" in the table to turn the functionality on or off.

If the new competitor price is affordable after applying Force Margin Check, the engine will find the corresponding competitor name again and calculate the total of skipped competitors counting from the old competitor price to the new one. Finding the corresponding competitor name is not relevant for the "Price Position" mode. In such case, the lowest affordable price will override the current price if in range of available prices.

**Note:** Order of operations is:

1. Price calculation by selected mode
2. Margin check
3. Reposition

#### Input Parameters

Input	Type	Description
Competitor and Prices	List	List with prices and competitor name from competitors that should used for processing.
Minimum Margin Price	BigDecimal	Price used for affordability check, nullable.

#### Additional Engine Configuration

This engine needs an additional simple configuration Price Parameter where all Configuration Options are present as keys.

Configuration Option	Expected value	Description
Competitor Position	Allowed values:	Selects the target competitor to compare to. Case insensitive.

	<ul style="list-style-type: none"> <li>• <code>min</code> - Selects the competitor with the lowest price.</li> <li>• <code>max</code> - Select the competitor with the highest price.</li> <li>• <code>min + x</code> - Selects the competitor with the lowest price and adjusts the position by X.</li> <li>• <code>max - x</code> - Selects the competitor with the highest price and adjusts the position by X.</li> <li>• <code>40%</code> - Selects the competitor whose position is at 40<sup>th</sup> percentile of the whole competitor range.</li> </ul>	
Price Position	Value in range: 0% - 100%	Directly selects the competitor price based on the percentage provided and the range of competitor prices.
Repositioning %	Value in range: 0% - 100%	Adjusts the selected competitor price by the given percentage.
Repositioning Abs	Absolute value. Can be negative.	Adjusts the selected competitor price by the given absolute value.
Force Margin Check	Allowed values: <ul style="list-style-type: none"> <li>• Yes</li> <li>• No</li> </ul>	Checks whether the selected competitor price is affordable based on Minimum Margin Price. If it is not, the next competitor price / affordable price is selected until an affordable price is found. If no such price is found, the exception is thrown.

#### Relevant Competitors Definition

You can decide if you want to use all existing competition data, or if you want to define relevant competitors. Definition of relevant competitors can be done on the Lookup Key Level. We support definition of a list of competitors. You can decide if these competitors should be used or excluded.

You can define relevant competitors in PP "RelevantCompetitionData". It has to be filled as followed:

Configuration Option	Expected Value	Description
Lookup Keys	Values of the Lookup Key	Keys in PP are the selected Lookup keys.
Relevant Competitors	"yes" or "no"	<ul style="list-style-type: none"> <li>• yes - relevant competitors are defined</li> </ul>

		<ul style="list-style-type: none"> <li>no - the list of competitors is excluded</li> </ul>
Competitor #1 ... Competitor #29	Name of Competitor	You can define up to 29 competitors to be considered as relevant or excluded.

Default Strategy Calculation Parameters

COMPETITOR\_PRICES (or RELEVANT\_COMPETITOR\_PRICES), MINIMUM\_MARGIN\_PRICE

## Kit Engine

Kit Engine calculates a price for a given product based on subcomponents and quantities defined in a standard BOM Data table.

A Kit Price is a sum of all of its subcomponent prices multiplied by provided quantities. There is no limit on how many levels of "subcomponent of subcomponent" are defined. If more than one level is present, we sum all the prices "at the bottom of the tree" using proper quantity factors.

The engine runs a cycle detection algorithm on the input BOM data. It throws an exception if a cycle is found.

This engine works properly only when all connected products are added to the same Price List or Live Price Grid.

### Important notes:

- This engine uses the "Dirty Run" functionality of Pricefx. It means that you should not mix products that are used by other engines that mark items as dirty (e.g. Anchor Engine) in one PL/LPG. It can lead to undefined behavior and wrong results.
- Because this engine doesn't return a price after the first calculation, if it is used as Base Strategy, it will not be shown in the "Prices" popup. The strategy will appear only when the second calculation run completes.
- If the used Price List or Live Price Grid are of the Matrix type, the engine assumes that the secondary key is used for storing the volume information and will search for a subcomponent product with the secondary key equaling to 1.

Input Parameters

Input	Type	Description
SKU	String	SKU of the calculated product
Bom List	List	BOM List for the currently calculated product as returned by <code>api.bomList()</code> or in the same format.
Final List Price Element Name	String	Name of the element that keeps the Final List Price. It is used for subcomponent price lookups if the calculation is in the Net mode.

Final Price Element Name	String	Name of the element that has the Final Price. It is used for subcomponent price lookups if the Final List Price element is empty. It usually happens during Gross calculation.
--------------------------	--------	--

Additional Engine Configuration

This engine does not have any additional configuration.

Default Strategy Calculation Parameters

SKU , BOM\_LIST , FINAL\_LIST\_PRICE\_ELEMENT\_NAME , FINAL\_PRICE\_ELEMENT\_NAME

**Lookup Engine**

Lookup Engine can be used for any strategy that has a static price that needs to be fetched from an existing table, e.g. Promotion Price or Recommended Retail Price.

It utilizes additional filtering based on:

- Target Date - Applied to "Valid From Field"/"Valid To Field" fields from the input configuration.
- Additional Filter Values - Passed values are OR'ed together and applied to the "Additional Filter Field" field from the input configuration. By default in the Price Setting package it uses all not-null.

At the end, we get a result for a given SKU, within the given validity dates and matching additional filter.

Input Parameters

Input	Type	Description
SKU	String	SKU of the calculated product.
Target Date	Date	Date used for the ValidFrom /ValidTo filtering.
Additional Filter Values	List	Used to pass values that will be used as additional filters. Empty by default.

Additional Engine Configuration

This engine needs an additional simple configuration Price Parameter where all Configuration Options are present as keys.

Configuration Option	Expected value	Description
Source Type	Allowed values: <ul style="list-style-type: none"> <li>• PX</li> <li>• PP</li> </ul>	Defines where the data is kept. For the PP type the engine expects a MATRIX table with SKU in the "name" column.
Source Table	ExampleTableName	Name of the data table.
Source Field	ExampleSourceField	Name of the column that contains the price within the table.

Valid From Field	ExampleFromColumn	Name of the Date type field.
Valid To Field	ExampleToColumn	Name of the Date type field.
Additional Filter Field	ExampleAdditionalFilterColumn	Name of the additional filter column. It can be any string that can match some data passed in the Additional Filter Values list.

Default Strategy Calculation Parameters

SKU , TARGET\_DATE , DEPENDENCY\_INFORMATION\_VALUES

## Net Engine

Net Engine calculates a gross price of a product based on a specific “pocket price” and discounts. The pocket price is always looked up using the Lookup Engine, so what this engine does is basically reverting any discounts that were applied to it.

Input Parameters

Input	Type	Description
SKU	String	SKU of the calculated product.
Target Date	Date	Date used for the ValidFrom/ValidTo filtering.
Discounts	List	Discounts that you want to “reverse”.
Additional Filter Values	List	Used to pass values that will be used as additional filters. Empty by default.

Additional Engine Configuration

This engine needs an additional simple configuration Price Parameter where all Configuration Options are present as keys.

Configuration Option	Expected value	Description
Source Type	Allowed values: <ul style="list-style-type: none"> <li>PX</li> <li>PP</li> </ul>	Defines where the data is kept.
Source Table	ExampleTableName	Name of the data table.
Source Field	ExampleSourceField	Name of the column that contains the price within the table.
Valid From Field	ExampleFromColumn	Name of the Date type field.
	ExampleToColumn	Name of the Date type field.

Valid To Field		
Additional Filter Field	ExampleAdditionalFilterColumn	Name of the additional filter column. It can be any string that matches some data passed in the Additional Filter Values list.

Default Strategy Calculation Parameters

SKU , TARGET\_DATE , DISCOUNTS , DEPENDENCY\_INFORMATION\_VALUES

## Custom Engines

In addition to using the predefined engines, you can also define your own strategies.

All you need to do is to place a function path to the function in some external Groovy Library (instead of an engine name in the PP StrategyDefinition), e.g. `libs.MyLib.MyElement.MyFunction`.

This function should return a calculated price or throw an exception. Its message will be shown in the PL /LPG.

Users may add their own parameters for the custom engine in the `AdditionalCalculatorParameters` element, according to the Groovy Documentation. It can be done by supplying one of these:

- Hardcoded value calculated earlier (`standardParameter`).
- Closure with a code which will be executed when the first element is empty (`optionalParameter`). This way this calculation can be lazy-initialized.

Possible return values of the engine-like function:

- Price as `BigDecimal`
- Thrown exception (will be handled and showed in Prices popup)
- Map with keys:
  - "price" - Price as `BigDecimal`.
  - "message" - Message shown in the Prices popup.
  - "messageType" - How message/price will be shown. Possible values are "Info", "Warning" and "Critical".

Example configuration of custom engines can be found at [https://pricefx.atlassian.net/wiki/spaces/ACCDEV/pages/2098036839/Recipe+Custom+Strategies?NO\\_SSR=1](https://pricefx.atlassian.net/wiki/spaces/ACCDEV/pages/2098036839/Recipe+Custom+Strategies?NO_SSR=1).

## Price Strategies

The main purpose of the Accelerate Price Setting Package is calculation and management of prices. That is why price calculation is one of the fundamentals in the package. Price Setting Package uses *Price Strategies* to calculate prices. From business perspective, price strategy represents the technical implementation of your *pricing rules*. When a price strategy is executed, it results in a price proposal for your product. You can have different price strategies for every product segment. We will calculate all of them and the one with the highest priority will end up as the final *price proposal* for your product.

Technically Price Setting Package uses [Calculation Engines](#) to calculate prices. A useable Pricing Strategy is a combination of Strategy Engine (some Groovy code) and [StrategyDefinition PP](#) (Price Parameter table combining the Strategy Engine with additional configuration parameters).

Pre-configured Price Strategies

The Package comes with the following pre-configured Price Strategies:

--	--	--

Price Strategy	How it calculates prices	What data is used
Minimum Competition Based Price	<p>The minimum competition price. You can configure if you want to:</p> <ul style="list-style-type: none"> <li>• directly map to the competitor price</li> <li>or</li> <li>• reposition against it (using relative or absolute values).</li> </ul>	<p>By default all competition data for the SKU is used. You can limit it to only relevant competition data. The minimum margin and cost are passed to the engine, so you can configure it to skip competitors which you cannot afford to position against.</p>
Average Competition Based Price	<p>The average competition price. You can configure if you want to:</p> <ul style="list-style-type: none"> <li>• directly map to the competitor price</li> <li>or</li> <li>• reposition against it (using relative or absolute values).</li> </ul>	
Maximum Competition Based Price	<p>The maximum competition price. You can configure if you want to:</p> <ul style="list-style-type: none"> <li>• directly map to the competitor price</li> <li>or</li> <li>• reposition against it (using relative or absolute values).</li> </ul>	
Recommended Retail Price	<p>Recommended retail price coming from an external source.</p>	<p>Lookup in Product Extension with a list of Recommended Retail Prices.</p>
Cost+	<p>Calculation of Cost+ Price. In the provided example it uses the relative plus factor (percentage) and applies it to the given cost base. It is possible to change it to an absolute value.</p>	<p>Product cost (or complex cost types), defined "plus" with absolute or relative values.</p>
Price Increase	<p>Increase of the previous price by a relative (percentage) factor. It is possible to change it to an absolute factor.</p>	<p>Actual price of the product and defined "increase" in absolute or relative values.</p>
Kit Pricing	<p>Kit Pricing calculates the price of a kit based on the prices of the sub-components. All of the sub-components have to be in the same PL/LPG.</p>	<p>BoM (Bill of Material) data to define sub-component relations.</p>
Attribute	<p>Attribute Based Pricing prices the products based on "value" of some</p>	<p>Product reference to define the "base product" for a special product.</p>

Based Pricing	product attributes. It takes the price from a defined reference product, and performs arithmetic operations (+, -, *, /) based on a defined formula and the values of some product attributes (e. g. "red" will result in + 3€, price will be multiplied by size, ...). These values can be direct numeric values (size, ...) or discrete values with assigned price impact.	<p>You will also need:</p> <ul style="list-style-type: none"> <li>• List of attributes that should be considered in the price calculation</li> <li>• Translation of discrete attributes or ranges of numerical values to price impact values (if you have such)</li> <li>• Formula to calculate the result price based on reference price and the dedicated price impacts</li> </ul>
---------------	--	--

### Strategy Engines

The described pre-configured examples are provided with the package.

If you need slightly different price strategies, please check how the strategy engines can be used and configured. They allow you to create various price strategies on your own. For details on the engines see:

- [Adjustment Engine](#)
- [Attribute Based Engine](#)
- [Anchor Engine](#)
- [Competition Engine](#)
- [Kit Engine](#)
- [Lookup Engine](#)
- [Net Engine](#)
- [Custom Engines](#)

### Custom Strategies

When you need some other specific rule to calculate your price, you can "plug in" your own [Custom Engines](#). A Custom Engine is basically some capsulated function that can be easily connected to the Accelerate Price Setting Package. For more technical details see [Recipe: Custom Strategies](#). You can use the out-of-the-box engines and wrap them in your own engine or start completely from the scratch.

### Strategy Importance

As you can see in [Price Strategies](#), the Accelerate Price Setting Package can calculate different prices based on different pricing rules. You can define many pricing strategies; for each of them you have the following configuration options in [StrategyDefinition PP](#):

- 'Level' describes where the definition is valid - for a dependent or independent price list. If you want a strategy to be valid in both scenarios, you need to create two entries.
- 'Overridable' describes if this strategy can be manually overridden by selecting other strategy or a manual price or using exception table.
- The remaining settings are used to order price strategies.

### How Independent and Dependent Levels Are Calculated

Generally there is the following mechanism to calculate the prices. It is different for Independent and Dependent calculations.

#### Independent Price List Calculations

Independent price list:

- Calculates base strategies.

- Calculates standard strategies.
- Removes base strategies which returned no price.

The highest level strategy will be used as price proposal. All other strategies will be available in the “Prices” pop-up and in the strategy selection drop-down (when allowed, check [PSP Override Module](#)).

#### Dependent Price List Calculations

Dependent price list is more complex, since it has several configuration options which tweak the strategies order:

- The first strategy is “Independent Level Adjusted Price”, which is a Final Price from the independent price list for a given product adjusted by the dependency adjustment. It can be overridden by setting “No” for the “Prioritize Independent Level Price” column in the [Strategy Selection Lookup](#) PP. In this case, “Independent Level Adjusted Price” will be put at the end of strategies. It is a configuration on the product level.
- Base and standard product prices are calculated. However, they come in pairs with independent level adjusted prices, if the same strategy was calculated for a given product in the independent price list.
- Prices from the independent price list can be ignored on the dependent level by setting “Independent Level Only” to “Yes” in the [StrategyDefinition PP](#).
- Dependent prices come in pairs with independent level adjusted prices (with dependent before independent), unless “Independent Level Priority” is set in the [StrategyDefinition PP](#). Independent level priority is taken into consideration only when the strategy is defined for both dependent and independent level. It has to be set by an entry on the independent level.

#### ✔ Note:

- When strategy is calculated on the dependent level, it will use its local available data (as competition data, cost, ...).
- When strategy is taken from the independent level, it will take the independent level price and apply markup factor.

So you have to be careful to configure it correctly. For example, when you have Cost+ pricing strategy both for dependent and independent levels and you have the same cost, it will result in two different prices. One freshly calculated and one taken from the independent level and with an applied markup factor - which might seem unexpected.


#### Summary

These are your options to influence the Independent/Dependent behavior regarding importance of pricing strategies.

Flag	Where you find the flag	What it does
Prioritize Independent Level Price	PP Strategy Selection	Default is “Yes”. When you change to “No”, this will force the system to put the “Independent Level Adjusted Price” (= final approved price from independent level with the markup) to the end of the priority list.
Level	PP Strategy Definition	You can have “Independent” and “Dependent” as level in the definition of strategies. When you want to calculate them on both levels, you have

	to add them twice. Be aware that “calculate” means, that they are freshly calculated on the independent level. When you only want to take some price from the independent level and add a markup factor, you do not have to configure the strategy for the dependent level.
Independent Level Only	You can only set up this flag for “Level” = “Independent”. When this is done like this, it will prevent the inheritance of this independent level price to the dependent level. So with this you force a price not to be taken over to the independent level and applied with the markup factor.
Independent Level Priority	When one strategy is calculated both on the independent and dependent levels, it will appear twice in the dependent level: <ol style="list-style-type: none"> <li>1. (Re)calculated on the dependent level base.</li> <li>2. Taken from the independent level and applied with markup factor.</li> </ol>

## Product Segmentation

 Product Segmentation controls how some Price Parameters will be generated during the deployment. It means that changes to this configuration require additional administrative actions described in [Changing product segmentation](#).

We support a “general” lookup key that is applicable for every lookup where you do not define a specific one. You can define a specific set of lookup keys for different tables and features in Price Setting Package during deployment.

There is the option to define other specific lookup keys per feature. It is possible to define the lookup keys for following features:

Feature	Description
Fallback	Used whenever there is no specific lookup key per feature.
StrategySelection	Selection of the strategies and importance.
MinMargin	The minimum margin used for warnings and margin checks.
DependencyLevelAdjustment	A markup factor between the Independent Price List and the Dependent one.
VolumeBreakdown	Quantities and volume discounts for price lists.
CostPlus	The factor for the Cost+ Price strategy that is applied to the cost base to get a price.

PriceIncrease	The factor for (periodic) price increase that is applied to the old price to get a new increased price.
AdditionalDiscount	A predicted additional discount. This is used in Target Price Strategy to anticipate influence of additional On-Invoice and Off-Invoice Price conditions.
BaseStrategySelection	Selection of "Base Strategies". Base Strategies are usually defined on more generic level to have some basic pricing rules across the complete product portfolio.
AdjustedPriceCorridor	A corridor used for price harmonization checks. It is used to check how strong the consistency in the general pricing rules is. The smaller this KPI is, the more the Dependent Pricing is aligned with the overall rules set. When strictly following the Independent Level Prices and the defined markup factor, it is zero.
ListPriceCorridor	A corridor used for price harmonization checks. This is used to check the harmonization of the prices themselves between Independent Level Price in Dependent Level Price. When AdjustedPriceCorridor is zero, this will exactly mirror the markup factor between the two.
RelevantCompetitionData	Parameter to define the relevant competitors. You can decide if your competition based strategies will use all competition data or only the set of relevant competition data.
CostSelection	When you have more than one cost type (e.g. <i>Cost with freight, Average Warehouse Cost, ...</i> ) you can decide per product segment which of them is used (for calculation of margin, for Cost+ pricing strategy, ...).
Discount	The discount you have in your discount structure. It is used in Gross/Net mode to calculate the net price based on the calculated List Price.

Each of the lookups will have generated multiple tables, based on dependency hierarchy. For details see [Dependent Price Lists and Data Fallbacks](#).

To learn more about Hierarchical Lookups, visit [Hierarchical Lookups](#).

**i** Before deployment it is important to consider what features will be used. For every feature you have to think about the granularity which will be required for data later. This is generally a business decision.

## Price Setting Package Administration Procedures

A list of step-by-step tested useful procedures:

- [Add New Dependency Level](#)
- [Rename Dependency Level Names](#)
- [Adjustments after Changing Price Setting Level](#)
- [Change Product Segmentation](#)

## Add New Dependency Level


### Business Requirements

- Decide:
  - Is this going to be a “Virtual Dependency Level” (used only as a fallback for other dependency levels which are lower in hierarchy) or a standard one?
  - Is this going to be “Independent” or “Dependent on already existing dependency” Level?
    - If Dependent, should it be in master relation or HQ relation?
  - Is this going to be “Complete Dependency Level” (no fallback to more general Dependency Levels, nor general fallback)?
- Prepare data for Dependency Level (or agree to utilize a fallback to some of them).
- Prepare Hierarchical Config for Dependency Level (or agree to utilize a fallback to some of them).

### New Dependency Level Check List

- Add a new entry to Dependency Configuration. All values will come from business requirements described above.
  - Remember to enter PL/PG ID of Master Dependency Level, not just Dependency Level just being added.
  - Remember that Preferences are used for Dependency Mapping [DependencyMappingConfig PP](#). PSP does not support different Dependency Mapping per Dependency Level, so it needs to be similar to data entered for other Dependency Levels.
- Run bootstrapping, according to this guide: [Changing product segmentation](#)
- Enter data in newly created Hierarchical tables (if needed, fallback might be utilized if Dependency Level is not complete - a general fallback is always utilized). List of all Hierarchical tables: [Hierarchical Lookups](#).
- Enter data for each Data Lookup (if needed, fallback/general fallback might be utilized if Dependency Level is not complete). List of all Data Lookups: [Data Lookups](#)
- Create a new PL/PG, according to the second part of this section: <https://pricefx.atlassian.net/wiki/spaces/ACCDEV/pages/2321809516/How+to+Deploy+Price+Setting+Accelerator#Configuration-and-Price-List%2FGrid-Creation>

### Rename Dependency Level Names

 It is not recommended to make these changes when the system is running in a production environment if you need to have fully functional historical calculation data.

Sometimes it is necessary to rename Dependency Level Name defined in [DependencyConfiguration PP](#). To do that, you have to:

1. Rename it in [DependencyConfiguration PP](#) - in the Dependency Level Name column.
2. During the bootstrapping process, we automatically generate PP names based on a new name. Search for all PPs starting with the old name and change it to new one.
3. Rename it in [PriceSettingLevel PP](#).
4. Delete all active LPGs and PLs that were created for this Dependency Level Name and create new ones with the new name selected in the input configuration. Because the input changed, old approved PLs will **not be considered as the same dependency level** after the renaming.

### Adjustments after Changing Price Setting Level

#### Issues

## Usage Clarity

Changing Price Setting Level will result in starting/stopping calculation of conditional elements. Calculation will always be performed with same result, but there might be some columns missing or appearing and never changing their values.

## Changing Final Price

Not only results will become obsolete at the moment of changing Price Setting Level. Running calculation of dependent Dependency Level will result in issues with reading Final Prices of master Dependency Level calculation.

## Solutions

1. If possible, create a new PL/PG for all Dependency Levels and update values in Dependency Configuration (PG/PL ID).
2. If not possible, rerun the whole PL/PG. Hide "zombie fields" (never updated) by preferences.
3. Rerun all PLs/PGs starting from Independent Dependency Level and going through hierarchy tree.

## Change Product Segmentation

### Technical Requirements for Bootstrapping

Detailed description of requirements can be found at <https://pricefx.atlassian.net/wiki/spaces/ACCDEV/pages/3252061372/Hierarchical+Lookups#Configuration-Of-Bootstrapping>.

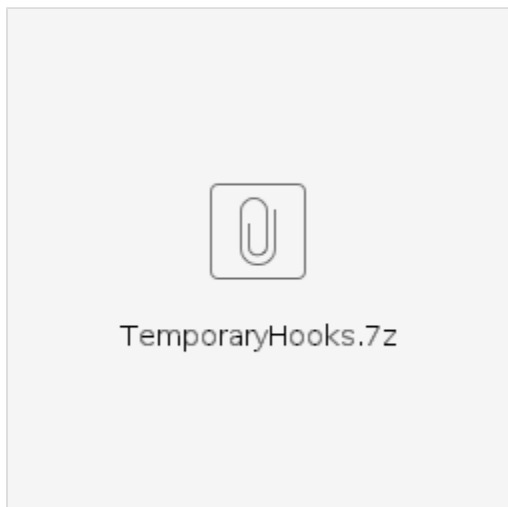
### Change Segmentation Check List

- Configuration needs to be updated:
  - [PriceSettingDimensions PP](#)
  - [DependencyConfiguration PP](#)
- Choose one of these two options:
  - Run "Price Setting Package - Upgrade" in PlatformManager Marketplace if you have the newest version of Accelerator. If you do not have the newest version, but you want to upgrade it anyway, follow [How to Upgrade Price Setting Accelerator](#).
  - Do it manually:
    - Deploy all Temporary Hooks. These are added to this page as attachments. You can change their folder, if you want new PPs to be created in another place.
    - Check for any changes in "CF\_BuildPricingTables" in [Accelerator repository](#), if there was any fixed bug.
    - Run logic named "CF\_BuildPricingTables". It can be done through CF or as a debug (with "Allow object modification" checked in).
- Clean up.
  - Remove Price Parameters which are not used anymore.
    - Price Parameters from old Dependency Configuration are not removed automatically.
    - There are a few restrictions on changing Dimensions, so always check if column names have been updated correctly. If the number of keys decreased and one of the keys moved to a previous position, then PP could become corrupted. Then it needs to be removed and added manually.
  - Add values to new Hierarchical Tables.

### Troubleshooting

- If you run bootstrapping during calculation, calculation is almost sure to fail. Most configurations are kept in cache for all products in PL/PG. Be sure to rerun after bootstrapping is finished.

- If bootstrapping did not work properly for any reason, a bug should be reported. However, wrong generation of Hierarchical Price Parameters will damage the whole Accelerator. If you are going to fix any issues manually, before Accelerator Team can respond:
  - Make sure that you have the newest, not modified version of CF\_BuildPricingTables.
  - Remember to deploy TemporaryHooks every time before you run bootstrapping.
  - Do not be afraid to run bootstrapping multiple times. The worst thing that can happen is that config is lost (which should be backed up anyway) or there are more leftover PPs on the partition (if the config is different every time).
  - Bootstrapping just creates Hierarchical Price Parameters (and removes Temporary Hooks). It can be done manually, but it will be time consuming. Config describing how data is read is in Price Parameters [PriceSettingDimensions](#) and [DependencyConfiguration](#).



## Price Setting Modules

Modularization is one of main concepts in Price Setting Package. It means that the package is split into a single required Core Module and multiple independent feature modules. This separation allows the package to stay fairly simple for small installations, while also allowing for more complex feature rich configurations.

Since modules are mostly independent, most of them have warnings and errors on the module level. In case something goes wrong in the module, the rest of them will still work.

Modules can be configured through our [Price Setting Package Configuration Wizard](#) or manually by turning them on in [PriceSettingModules PP](#) and configuring according to individual module's configuration page.

Available modules:

Module Name	Description	Configuration key name
<a href="#">Core Elements</a>	Basic Module of Price Setting Accelerator. In the module you can do the basic configuration of the package.	
<a href="#">Transaction</a>	Displays transaction and forecast data about products.	PSP_TRANSACTION_MODULE

	Stock data is independent from transactions, but calculation of StockCoverDays is dependent on this module.	
Net Price	Allows you to calculate a net price (with a proper discount taken into consideration). This is usually used in B2B(2C) Business.	PSP_NET_PRICE_MODULE
Overrides	Handles exceptions in pricing. It allows you to manually override product prices in the Price List / Price Grid or store exceptions per SKU.	PSP_OVERRIDES_MODULE
Price Checks	Checks if the user margin is within a suitable range and if not, it issues alerts. Also, for dependent price lists, it checks if the difference between a dependent and independent price is within a suitable range.	PSP_PRICE_CHECKS_MODULE
Price Flexibility	Provides integration with <a href="#">Price Flexibility Package</a> . It adds a new element to the independent price grid which describes why a product has been automatically added to a price grid.	PSP_PRICE_FLEXIBILITY_MODULE
Product Competition	Gathers and displays product competition data. This can be used for any competition based strategy.	PSP_PRODUCT_COMPETITION_MODULE
Strategy Conditions	Performs additional checks if prices meet certain conditions. Strategies can be skipped or used as fallback to ensure proper pricing rules.	PSP_STRATEGY_CONDITION_MODULE
Rounding Rules	Rounds prices to user friendly values.	PSP_ROUNDING_RULES_MODULE
Advanced Cost	Calculates additional cost types. These will be used for pricing strategies and margin calculations.	PSP_ADVANCED_COST

## PSP Core Elements

Price Setting Package Core Elements consist of:

- [PSP Cost Element](#)
- [PSP Actual Price Element](#)
- [PSP Stock Element](#)

### PSP Cost Element

#### Lookup

1. Search for records of the current dependency level in the selected source table. If none is found, use the dependency fallback mechanism.
  - Details about the dependency fallback mechanism can be found at [Dependent Price Lists and Data Fallbacks](#).
  - The selected source table depends on the dependency mapping configuration.
2. Filter records with current valid time configuration.

#### Calculation with Cost

1. Convert the cost currency to the current currency.
  - Details about currency conversion can be found at [ExchangeRates PP](#)
2. Perform the calculation based on the selected price strategies.

#### Configuration

1. Set the data source configuration in the PriceSettingConfig PP table. Details can be found at [Cost Config \(PriceSettingConfig\)](#).
2. Set the dependency mapping in the DependencyMapping PP table with the key `Cost`. Details can be found at [DependencyMappingConfig PP](#).

### PSP Actual Price Element

#### Lookup

##### *Product Extension Source Type*

1. Search for records of the current dependency level in the selected source table. If none is found, use the dependency fallback mechanism.
  - Details about the dependency fallback mechanism can be found at [Dependent Price Lists and Data Fallbacks](#).
  - The selected source table depends on the dependency mapping configuration. Details can be found at [DependencyMappingConfig PP](#).
2. Filter records with current valid time configuration.

##### *Price List Source Type*

1. Search for the latest approved price list which contains the record for the current SKU. The price list must also have the same dependency level name and calculation logic name as the current PL/PG.
2. Get the final price of the found item.

##### *Live Price Grid Source Type*

1. Search in the current PG for the latest approved record of the current SKU.
2. Get the final list price of the found item. If the final list price is not available, get the final price.

#### Calculation with Actual Price

1. Convert the price currency to the current currency.  
Details about currency conversion can be found at [ExchangeRates PP](#).
2. Perform the calculation based on the selected price strategies.

#### Configuration

1. Set the data source configuration in the PriceSettingConfig PP table. Details can be found at [Actual Price Config \(PriceSettingConfig\)](#).
2. Set the dependency mapping in the DependencyMapping PP table with the key `Actual Price`. Details can be found at [DependencyMappingConfig PP](#).

### PSP Stock Element

#### Lookup

1. Search for records of the current dependency level in the selected source table. If none is found, use the dependency fallback mechanism.
  - Details about the dependency fallback mechanism can be found at [Dependent Price Lists and Data Fallbacks](#).
  - The selected source table depends on the dependency mapping configuration.
2. Filter records with current valid time configuration.

#### Calculation with Stock

Used to calculate Stock Cover Days.

#### Configuration

1. Set the data source configuration in the PriceSettingConfig PP table. Details can be found at [Stock Lookup](#).
2. Set the dependency mapping in the DependencyMapping PP table with the key `Stock`. Details can be found at [DependencyMappingConfig PP](#).

### PSP Override Module

The Override module allows you to create custom price behavior that does not follow the default rules.

#### Visible Elements

Visible elements of the Override module are `Override Price`, `Override Reason`, `Price Selector`, and `Exceptions`. These elements override the calculated prices.

Element name	Label	Independent PL/PG	Dependent PL/PG	Description
PriceSelector	Price Selector	Yes	Yes	Dropdown list of calculated prices to choose from.
ManualPrice	Override Price	Yes	Yes	To enter a price manually.

ManualPriceReason	Override Reason	Yes	Yes	To enter a comment manually.
Exceptions	Exceptions	Yes	Yes	Information about the override values (if any).

## Override Levels

**⚠** Changing this configuration adds or hides some visible fields and because of the way Pricefx treats such changes, all used PLs and PGs should be recreated from scratch. Otherwise there will be "zombie columns" which will make the impression that exceptions do not work correctly.

- The current line (LineLevel) - Can only set the override values on the current SKU line in PL/PG. The ExceptionTable values are excluded.
- Configuration tables (ExceptionTable) - Can only set the override values through the configuration tables. The LineLevel values are excluded.
- Both (Yes) - Uses both Current line and Exception tables methods.
- None (No) - Override is not allowed.

More details about override levels can be found at [Exceptions and Manual Override Allowance Config \(PriceSettingConfig\)](#).

## Override Mechanism

Depending on the Override level configuration, the user can override a product price or a product strategy in different ways. The `Price Selector` element contains the calculated prices based on specified strategies and exceptional values.

- To set up a strategy, see [StrategyDefinition PP](#).
- For a list of built-in calculation engines, see [Calculation Engines](#).
- To specify the strategies used to calculate the price for an SKU, see **//TODO**

Type	Usage	Current line (Manual override)	Configuration tables (Exception table)	Both	None
Price	To override an SKU price with a specific price	Type the price in the <code>Manual Price</code> field	Set the price in a configuration table	Can do both Current line and Exception tables methods	Override is not allowed
Strategy	To use a specific strategy in the calculated prices as the selected strategy	Select a strategy from the <code>Price Selector</code> dropdown list	Set the strategy in a configuration table	Can do both Current line and Exception tables methods	Override is not allowed

## Override Order (Highest to Lowest)

Order	Name	Price to calculation	Price Decision	Necessary Action
1	Manual Price Override	Price from the <code>Manual Price</code> field	Default comment is inserted if none given. It can be manually overridden.	Type a price in the <code>Manual Price</code> field

2	Manual Strategy Override	Price from PriceSelector	Default exception message with the name of the price strategy chosen in the exception.	Choose a strategy from the PriceSelector drop-down
3	Price Exception	Price from an exception table	Default exception table message.	Set up an exception for the product in table
4	Strategy Exception	Price from the price strategy chosen in the exception	Default exception message with the name of the price strategy chosen in the exception.	Set up an exception for the product in table
5	Normal	Price from the first strategy in the calculated prices (first/bold row in the Prices popup)	Name of the price strategy.	N/A

Some levels of this hierarchy can be skipped by changing the Manual Override Allowance configuration. For example, when setting the Independent Manual Override for a price to "ExceptionTable", it will disable the Manual Price Override from this hierarchy.

## Exception Data Sources

### Price Exception

#### Lookup

1. Search for records of the current dependency level in the selected source table. If none is found, use the dependency fallback mechanism.
  - Details about the dependency fallback mechanism can be found at [Dependent Price Lists and Data Fallbacks](#).
  - The selected source table depends on the dependency mapping configuration.
2. Filter records with current valid time configuration.

#### Calculation with Price Exception

1. Convert the exception value currency to the current currency.
  - Details about currency conversion can be found at [ExchangeRates PP](#).
2. The exception value is in the Price Selector element for selection. If this is the default final price or not depends on the level configuration and override orders.

### Strategy Exception

#### Lookup

1. Search for records of the current dependency level in the selected source table. If none is found, use the dependency fallback mechanism.
  - Details about the dependency fallback mechanism can be found at [Dependent Price Lists and Data Fallbacks](#).
  - The selected source table depends on the dependency mapping configuration.
2. Filter records with current valid time configuration.

## Calculation with Strategy Exception

The exception value is in the `Price Selector` element for selection. If this is the default final price or not depends on the level configuration and override orders.

### Configuration

#### Set Module Status

Set the module status in the `PriceSettingModules PP` table whose the module name is `PSP_OVERRIDES_MODULE`.

#### Set Override Levels

1. Set the product price override level for independent pricing in the `PriceSettingConfig PP` table with the keys `Independent Manual Override Allowance | Price`.
2. Set the product strategy override level for independent pricing in the `PriceSettingConfig PP` table with the keys `Independent Manual Override Allowance | Strategy`.
3. Set the product price override level for dependent pricing in the `PriceSettingConfig PP` table with the keys `Dependent Manual Override Allowance | Price`.
4. Set the product strategy override level for dependent pricing in the `PriceSettingConfig PP` table with the keys `Dependent Manual Override Allowance | Strategy`.

The override level options:

- Yes
- No
- LineLevel
- ExceptionTable

#### Set Price Exception Data Source

1. Set the data source configuration in the `PriceSettingConfig PP` table whose key is `Price Exception`. Details can be found at [Exception Lookup](#).
2. Set the dependency mapping in the `DependencyMapping PP` table whose key is `Price Exception`. Details can be found at [DependencyMappingConfig PP](#).

#### Set Strategy Exception Data Source

1. Set the data source configuration in the `PriceSettingConfig PP` table whose key is `Strategy Exception`. Details can be found at [Exception Lookup](#).
2. Set the dependency mapping in the `DependencyMapping PP` table whose key is `Strategy Exception`. Details can be found at [DependencyMappingConfig PP](#).

## PSP Rounding Rules Module

The Rounding module allows users to round prices to business-friendly values. Rounding is done by using a defined set of rules. This set can be expanded if needed.

### Note:

- Values from PSP Exception Module will not be rounded (manually overridden prices and Exception Prices).

s	Price Selector	Override Price	Override Reason	Exceptions	Fin
	PPP1400.00			Strategy Exception: MaxCompetitor	

- Default Manual Override of Pricefx will be rounded.

	Manual Override ▼	Actual Price Lo
.10	[manualResultPrice]	
.10		

- Only **List Price** is rounded for each calculated strategy. So when using the **Net Calculation Level**, it only rounds the Gross Price.

### Mechanism

This module rounds prices to business-friendly targets. Manually overridden prices and Exception Prices will not be rounded.

### Targets

- To49Cents: XXX.YYY => XXX.49
- To50Cents: XXX.YYY => XXX.50
- To95Cents: XXX.YYY => XXX.95
- To99Cents: XXX.YYY => XXX.99
- ToWhole: XXX.YYY => XXX
- To5Whole: XXX.YYY => XX5
- To49Whole: XXX.YYY => X49
- To99Whole: XXX.YYY => X99
- RawPrice\*: XXX.YYY => XXX.YY
- NoRounding: do not round

### Modes

- UP - Round away from zero.
- DOWN - Round towards zero.
- HALF\_UP - Round towards "nearest neighbor" unless both neighbors are equidistant, in which case round up.
- HALF\_DOWN - Round towards "nearest neighbor" unless both neighbors are equidistant, in which case round down.

More information about rounding mode definitions and examples can be found at [Rounding Modes](#).

### Configure Rules

In the rounding rules configuration table, specify these fields:

- From (must be PP key1) - The price to be rounded should be greater than or equal to this field value.
- To (must be PP key2) - The price to be rounded should be less than this field value.
- Dependency Mapping Field - The value to be used for dependency mapping. Not required if using the table mapping mode.
- Rounding Rule - The target to be applied.

- Rounding Mode - The mode to be applied.
- Valid From - Valid start date. Optional.
- Valid To - Valid end date. Optional.

Examples:

<input type="checkbox"/>	From	To	Country	Rounding Rule	Rounding Mode
<input type="checkbox"/>	5.00	10.00	Global	To49Cents	HALF_UP
<input type="checkbox"/>	10.00	20.00	Global	To50Cents	UP
<input type="checkbox"/>	20.00	30.00	Global	To95Cents	HALF_DOWN
<input type="checkbox"/>	30.00	40.00	Global	To99Cents	UP
<input type="checkbox"/>	40.00	49.00	Global	ToWhole	DOWN
<input type="checkbox"/>	50.00	55.00	Global	To99Whole	DOWN

### Rule Data Source

Lookup

1. Search for records of the current dependency level in the selected source table. If none is found, use the dependency fallback mechanism.
  - Details about the dependency fallback mechanism can be found at <https://pricefx.atlassian.net/wiki/spaces/ACCDEV/pages/2962817818/Dependent+Price+Lists+and+Data+Fallbacks#Data-Fallbacks>.
  - The selected source table depends on the dependency mapping configuration.
2. Filter records with current valid time configuration.

### Configuration

Set Module Status

Set the status in the PriceSettingModules PP table which the module name is PSP\_ROUNDING\_RULES\_MODULE

Set Rule Data Source

1. Set the data source configuration in the PriceSettingConfig PP table with the key Rounding Rules. Details can be found at [Rounding Rules Lookup](#).
2. Set the dependency mapping in the DependencyMapping PP table with the key Rounding. Details can be found at [DependencyMappingConfig PP](#).

### PSP Net Price Module

This module calculates net prices (with proper discounts taken into consideration). This is usually used in B2B or B2C businesses.

### Module Related Elements

Technical Name	Label		UI Visible	Output Type	Description
----------------	-------	--	------------	-------------	-------------

		Available in Independent PL/PG	Available in Dependent PL/PG			
NetPriceLevel		Yes	Yes	No	Int (bool)	Indicates if the net price will be calculated.
NetPrice	Net Price	Yes	Yes	When pricing mode is Gross/Net and the module is turned on	BigDecimal	List price with a discount applied.
Discount	Discount	Yes	Yes		BigDecimal	Discount (%) to be applied when transitioning from a gross to net price.
FinalListPrice	Final List Price	Yes	Yes		BigDecimal	See the Mechanism section.
FinalPrice	Final Price	Yes	Yes	Yes	BigDecimal	


### Net Price Module and Pricing Mode

Pricing mode	Module status	Visible	Final list price	Net price	Final price
Gross	On	No	Not available	Not available	Strategy calculated price
Gross	Off	No	Not available	Not available	Strategy calculated price
Gross/Net	On	Yes	Strategy calculated price	Final list price with Discount applied	Net price
Gross/Net	Off	No	Not available	Not available	Strategy calculated price

To set the pricing mode, see [PriceSettingLevel PP](#) and [Adjustments after changing Price Setting Level](#).

### Discount Data Sources

- Discount data tables - Used for the regular PL/PG calculation.  
Table name convention: <<dependency>>Discount.  
Examples: Discount, AsiaDiscount, GlobalDiscount,...
- Additional discount price parameter tables - Used for the Net engine.  
Table name convention: <<dependency>>AdditionalDiscount.  
Examples: AdditionalDiscount, AsiaAdditionalDiscount, GlobalAdditionalDiscount,...
- Lookup - Search for the table which matches the current dependency level. If none is found, use the dependency fallback mechanism.  
Details about the dependency fallback mechanism can be found at [Dependent Price Lists and Data Fallbacks](#).

 When using the dependency fallback mechanism for <<dependency>>Discount /<<dependency>>AdditionalDiscount, if a dependency level in the tree has a complete state Yes but the value cannot be found, it will get the value from the root table, which is the Discount table / AdditionalDiscount table.

## Net Price Module and Net Engine

The Net engine takes a target price and performs calculations to produce the net price. The data source for the `target price` is defined in the corresponding additional engine configuration table. For details see [Net Engine - Additional Engine Configuration](#).

When using the strategy running on the Net engine, the following mechanism is applied:

Net Price module	Discount (%)	Additional discount (%)	Strategy calculated price	Net price	Final price
On	Some value	Some value	$\text{target price} / (1 - \text{Additional Discount}) / (1 - \text{Discount})$	$\text{calculated price} * \text{discount}$	net price
On	Null	Null	target price	Null	Null
On	Some value	Null	$\text{target price} / (1 - \text{Discount})$	$\text{calculated price} * \text{discount}$	net price
On	Null	Some value	$\text{target price} / (1 - \text{Additional Discount})$	Null	Null
Off	N/A	Some value	$\text{target price} / (1 - \text{Additional Discount})$	N/A	calculated price
Off	N/A	Null	target price	N/A	calculated price

## Configuration

To turn the module on or off, update the module status in the `PriceSettingModules PP` table whose name is `PSP_NET_PRICE_MODULE`.

## PSP Transaction Module

This module displays transaction and forecast data about products. Stock data is independent from transactions, but the calculation of `StockCoverDays` depends on this module.

### Module Visible Elements

Visible elements of the Transaction module are `Sales Volume YTD`, `Turnover YTD`, `Last Year Sales Volume`, `Last Year Turnover`, `Last Period Volume`, `Last Period Turnover`, `Sales Volume Forecast`, and `Turnover Forecast`. These elements display transaction and forecast data of the product, which are sales data for the last year and forecast data for the next year.

Type	Technical Name	Label	Independent PL/PG	Dependent PL/PG	Description
Historical	SalesVolumeYTD	Sales Volume YTD	Yes	Yes	The sum of sales volume from the beginning of the current year to the calculation date.

	TurnoverYTD	Turnover YTD	Yes	Yes	The sum of turnover from the beginning of the current year to the calculation date.
	LastYearSalesVolume	Last Year Sales Volume	Yes	Yes	The sum of sales volume from the whole last year.
	LastYearTurnover	Last Year Turnover	Yes	Yes	The sum of turnover from the whole last year.
	LastPeriodVolume	Last Period Volume	Yes	Yes	The sum of sales volume in a specified time range in the past.
	LastPeriodTurnover	Last Period Turnover	Yes	Yes	The sum of turnover in a specified time range in the past.
Forecast	SalesVolumeForecast	Sales Volume Forecast	Yes	Yes	The sales volume forecast.
	TurnoverForecast	Turnover Forecast	Yes	Yes	The turnover forecast.

#### Last Period calculation

The last period configuration allows flexible data lookup. The time units used in this calculation (days, weeks, months, and years) are not days counting from the beginning. In other words, when specifying the last period time as 1 week, it does not mean taking 7 days from the calculation day backward. The calculation takes only periods that are finished and a week starts on Sunday, ends on Saturday.

For example, the calculation date is 22 September 2020, and the last period configuration "1 week". The result will be the data from 13 September to 19 September.

#### Forecast Types

Last year	Linear	Lookup
The sum of sales volume /turnover from the whole last year	$ytdData = \text{sum of sales volume/turnover from the beginning of the current year to the calculation date}$ $ytdDaysCount = \text{number of days from the beginning of the current year to the calculation date}$ $currentYearDaysCount = \text{number of days of the calculation year}$ $result = ytdData / ytdDaysCount * currentYearDaysCount$	The sum of sales volume /turnover from the beginning of the current year to the calculation date. The data is obtained from a defined data source.

To configure the forecast type, see [Forecast Config \(PriceSettingConfig\)](#).

#### Currency Exchange

Transaction data may point to Product Extensions, Datamart or Data Source. In the Price Setting Package, the PL/PG logic will ignore the "currency" field in Datamart source tables as the exchange rate has been applied during the data transition from Data Source to Datamart. The exchange rate is only applied when the Datamart's currency and the calculation's currency are different.

When the transaction data source points to Data Source, the "currency" field of the pointed Data Source table will be utilized. For details, see [Set up Currencies](#).

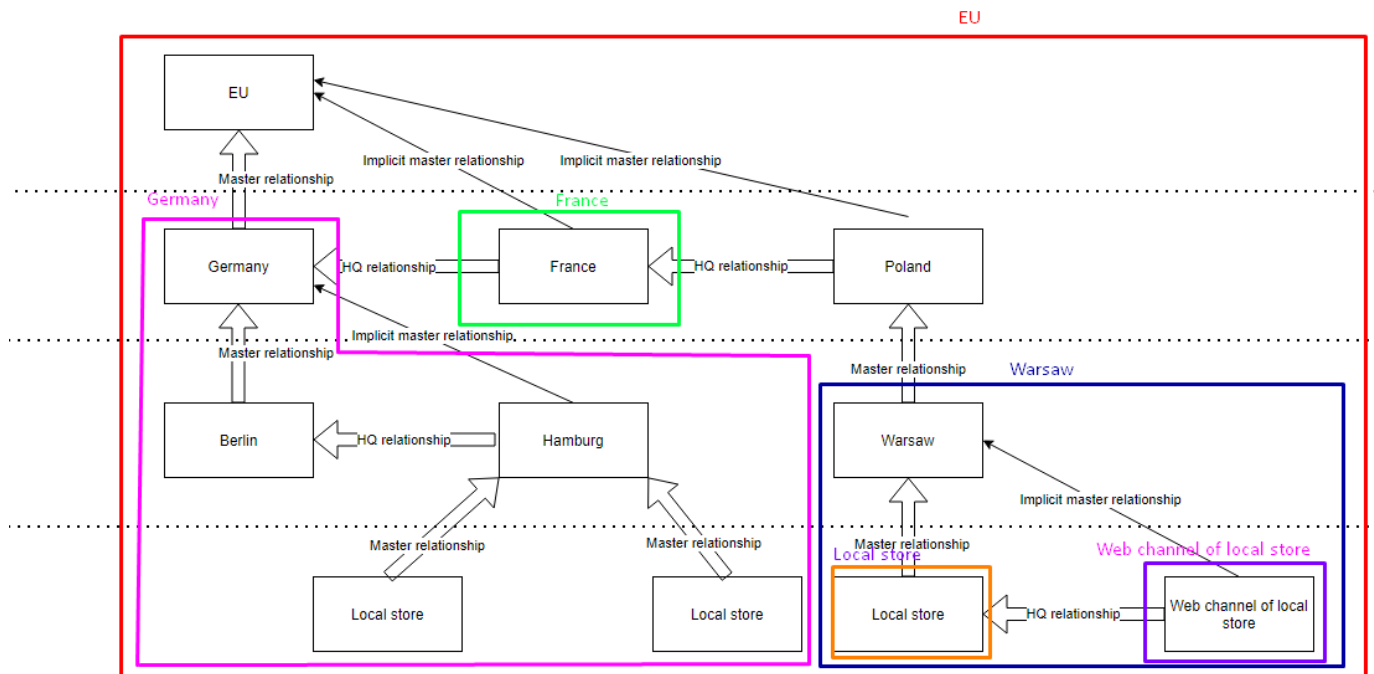
When the transaction data source points to Product Extensions, the currency is converted using the ExchangeRate PP. For details, see [ExchangeRates PP](#).

### Lookup

Transaction data lookups return aggregated data for the current and all children dependency levels and there is no fallback like other lookups. Transactions are records of anything being sold. If something has been sold in a city, it means that these transactions should also appear as being sold in the country.

If there is no dependency defined, e.g. a small company working only in a country, without changing their prices depending on the regions, it will grab all transactions.

Example:



It collects everything from Germany and levels defined as children, and children’s children, etc. In this example, Germany data = Germany + Berlin + Hamburg + Local store + Local store. It does not include nodes at the same level, which are France and Poland in this case.

### Configuration

#### Set Module Status

Set the module status in the PriceSettingModules PP table whose module name is PSP\_TRANSACTION\_MODULE

#### Set Transaction Data Source

- For transaction data source, set it in the PriceSettingConfig PP at the key `Transaction Source`. For details, see [Transaction Lookup](#).
- For transaction dependency mapping, set it in the DependencyMapping PP table at the key `Transaction`. Details can be found at [DependencyMappingConfig PP](#).

#### Configure Forecast Calculation

- For forecast setting, set it in the PriceSettingConfig PP at the key `Forecast`. For details, see [Forecast Config \(PriceSettingConfig\)](#).
- For transaction dependency mapping, set it in the DependencyMapping PP table at the key `Forecast`. Details can be found at [DependencyMappingConfig PP](#).

#### Configure Last Period Calculation

To configure the last period calculation, set it in the PriceSettingConfig PP at the key `Last Period Transaction`. For details, see [Last Period Config \(PriceSettingConfig\)](#).

#### Warning

**⚠** Technically, if there are too many rows (1 million by default) for a given batch of products (200 by default), the batch will be split in two, a warning will be raised and the SQL query will be executed again. If there are too many rows for only 1 SKU, then the Pricefx restriction has been met, an error will be raised and no transaction data will be read.

In Product Extensions, however, we do not expect to have tens of thousands of rows. We expect data to be pre-aggregated. Having a lot of rows in PX is possible, but counterintuitive. Users are expected to fill 3 more columns when they use Price Insights Dashboard. Those are Min, Max, and Avg turnovers of all data aggregated into that entry.

## PSP Price Checks Module

The Price Checks module verifies if a product margin is within a suitable range. For dependent price lists, it also checks the delta between the independent price and the dependent price in a dependent PL/PG.

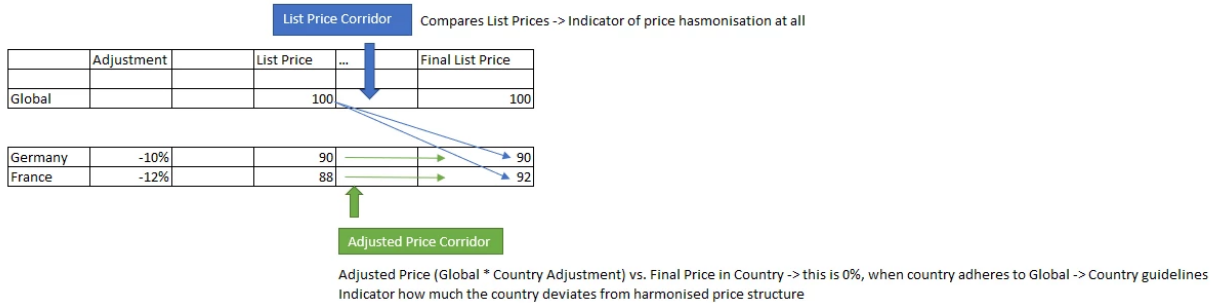
### Module Elements

Technical name	Label	Available in Independent PG/PL	Available in Dependent PG/PL	Description
MinimumMargin	Minimum Margin	Yes	Yes	The minimum margin specified for the product.
ListPriceCorridor	List Price Corridor	No	Yes	Delta between the independent price (*) and final list price.  On independent price, refer to Element Data Sources > List Price Corridor section below.
AdjustedPriceCorridor	Adjusted Price Corridor	No	Yes	Delta between the adjusted independent price and final list price.  On independent price, refer to Element Data Sources > Adjusted Price Corridor section below.
MinimumMarginPrice	Minimum Margin Price	Yes	Yes	A price calculated based on <code>MinimumMargin (%)</code> , <code>Cost</code> and <code>Discount (%)</code> .

#### Element Data

Values for price checks are configured in respective price parameter tables or calculated results.

- **Minimum Margin - MinMargin PP table**  
From table with naming convention: <<dependency>>MinMargin.  
Examples: MinMargin, AsiaMinMargin, GlobalMinMargin,...
- **List Price Corridor (Dependent PG/PL only)**  
The value is taken from the final list price of the referred independent PG/PL.  
To set the referred independent PG/PL for a dependency level, see [DependencyConfiguration PP](#).
- **Adjusted Price Corridor (Dependent PG/PL only)**  
The value is taken from the final list price of the referred independent PG/PL with the dependent adjustment applied.

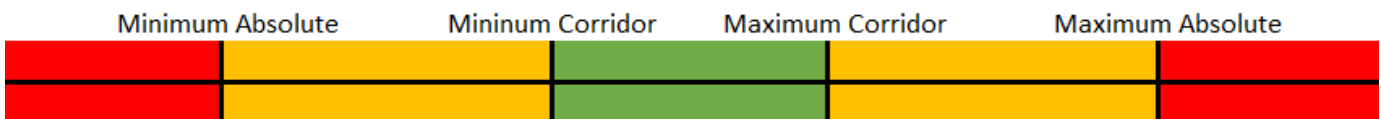


To set the dependent adjustment, fill the data in the corresponding <<dependency>>DependencyLevelAdjustment Price Parameter table.

- **Minimum Margin Price** - The value is calculated with the formula:
  - At Gross pricing mode: result = Cost / (1 - MinimumMargin)
  - At Gross/Net pricing mode: result = Cost / (1 - Discount - MinimumMargin + MinimumMargin \* Discount)

### Checking Mechanism

The List Price Corridor and Adjusted Price Corridor elements will be colored based on the corridor ranges which they fall into.



- When the element value is inside the Corridor range, it will be green.
  - Minimum Corridor <= value <= Maximum Corridor
- When the element value is inside the Absolute range, it will be yellow.
  - Minimum Absolute <= value < Minimum Corridor
  - Maximum Corridor < value <= Maximum Absolute
- When the element value is outside the Absolute range, it will be red.
  - value < Minimum Absolute
  - value > Maximum Absolute
- When the element value is zero, it will be blue.

## Configuration

### Set Module Status

Set the module status in the PriceSettingModules PP table whose module names is PSP\_PRICE\_CHECKS\_MODULE

### Set Minimum Margin

To specify the minimum margin for product segments, do it in the corresponding `<<dependency>>MinMargin` PP table.

Examples: MinMargin, AsiaMinMargin, GlobalMinMargin,...

### Set List Price Corridor

To validate the delta between the independent price and the final list price of a product, the corresponding boundaries are taken from the ListPriceCorridor PP table.

### Set Adjusted Price Corridor

To validate the delta between the adjusted independent price and the final list price of a product, the corresponding boundaries are taken from the AdjustedPriceCorridor PP table.

## PSP Strategy Conditions Module

The Strategy Conditions module lets users define special conditions based on which some prices will be ignored or taken with lower priority. Conditions are applied at the very end of the calculation, after exceptions. This feature does not, however, differentiate between independent and dependent prices in a dependent price list. Independent prices are fetched and paired with dependent ones after applying conditions.

To set a condition, there are these requirements: condition order, condition expression, and the rule to be applied.

### Condition Expression Syntax

In general, a condition expression has the form of `left-hand side operand operator right-hand side operand`. For example, `Cost+.margin<MINIMUM_MARGIN_PRICE`. A suggestion is to think of the left-hand side strategy as the main strategy for the given condition.

### Left-hand Side Operand

The left-hand side of the expression must always be a property of a given strategy. The supported properties are:

- Gross price of the strategy: `<some strategy name>.price`.
  - Example: `Cost+.price`, `RRP.price`, etc.
- The margin of the strategy: `<some strategy name>.margin`.
  - Example: `Cost+.margin`, `RRP.margin`, etc.

The strategy can be any calculated strategy or exception.

### Operator

Supported operators are: "`<`" (less than), "`>`" (greater than), "`=`" (equals).

## Right-hand Side Operand


The right-hand side of an expression may be either property of a strategy or PriceCalculator's parameter.

- Gross price of the strategy: `<some strategy name>.price`.
  - Example: `Cost+.price`, `RRP.price`, etc.
- The margin of the strategy: `<some strategy name>.margin`.
  - Example: `Cost+.margin`, `RRP.margin`, etc.
- PriceCalculator's parameter: `parameter name`
  - Example: `MINIMUM_MARGIN_PRICE`, `DISCOUNT`, etc.

In addition, the right-hand side may be modified by a certain multiplier. The syntax is to wrap the right-hand side operand into parenthesis "(...)", add an asterisk character and a string parsable to BigDecimal. Example of a condition expression using multiplier:

- `Cost+.price<(RRP.price * 2)`
- `Cost+.price<(MINIMUM_MARGIN_PRICE * 2)`

Also, custom values from `additionalParameters` and `additionalOptionalParameters` are accepted.

 The strategy name needs to be written explicitly. This is the case when using Dependent PL/PG and "Cost+ (Independent Type Strategy)" and "Cost+" are two different strategies.

## Rules

Name	Description	Syntax
Skip	Remove the strategy on the left-hand side from the strategy list.	(skip)
Fallback	Move the left-hand side strategy to the last position of the strategy list.	(fallback)
Move behind	Move a strategy after another strategy.	<code>\$strategyName1&lt;\$strategyName2</code> Example: <code>"Cost+&lt;RRP"</code> , which mean moving Cost+ strategy after RRP strategy.

## Wildcards

To manage a large number of strategies in one go, there is the wildcard "{any}". The condition with wildcard will be applied to every strategy and exception in the strategy list. At the place of a wildcard, strategy name will be inserted.

Example: `"{any}.margin<MINIMUM_MARGIN"`

## Configuration

### Set Module Status

Set the module status in the PriceSettingModules PP table where the module names is `PSP_STRATEGY_CONDITION_MODULE`.

### Set Condition

To set a strategy condition, set it in the StrategyConditions PP. To learn more about it, see [StrategyConditions PP](#).

## PSP Advanced Cost Module

This module provides additional cost strategies which are used for pricing strategies and margin calculations. Each cost strategy has its own dependency mapping configuration and data source configuration (including optional valid dates or currency).

### Mechanism

Similarly to pricing strategies, there can be many cost strategies. In a product segment, several cost strategies can be selected. The cost strategies will be calculated and their values are presented in the `Advanced Costs` element of the PL/PG. However, only the first valid value (top-down order) is used for margin calculation.

Example: If there are `Cost1=null`, `Cost2=45`, `Cost3=50`, then `Cost2` value will be used to calculate the margin.

### Advanced Cost and Simple Cost

By enabling the `AdvancedCost` module, the `Cost` configuration in `PriceSettingConfig` PP and `DependencyMappingConfig` will become unused and can be removed.

### Cost Aggregation Types

Currently, there are three types supported:

- `SINGLE` - Lookup for only one cost.
- `AVG` - Lookup for costs in the targeted Product Extension, then return the average of all found values.
- `SUM` - Lookup for costs in the targeted Product Extension, then return the sum of all found values.

### Cost Strategy and Pricing Strategy

#### Naming Convention

A cost strategy value can be passed to a pricing engine as a parameter. When defining a cost strategy, a proper engine suffix has to be provided. The base name of any engine parameter is always `"COST"`, so if a cost strategy has the `"_EXAMPLE"` engine suffix, then its engine parameter name is `"COST_EXAMPLE"`.

#### Usage

In terms of pricing engine parameters, only costs from the `CostSelection` PP for the current product are loaded + `"PRODUCT_COST"`.

Example:

In `CostTypeDefinition` PP, there are `COST1`, `COST2`, `COST3`, `COST4`, `COST5`, `COST6`.

In `CostSelection` PP, it is set to use `COST1`, `COST2`, `COST3`, `COST4`, `COST5`.

During the calculation process, `COST1`, `COST3`, and `COST4` were not calculated properly. It means that the product's cost will be `COST2` (first valid calculated cost value) and 3 parameters will be loaded to engines, which are `"PRODUCT_COST"`, `"COST2"`, and `"COST5"`. `"PRODUCT_COST"` represents the first valid cost and it has the same value as `COST2`.

## Configuration

### Set Module Status

Set the module status in the PriceSettingModules PP table whose module name is PSP\_ADVANCED\_COST

### Set Cost Strategy Definition

To set up a cost strategy, create a new data row in the CostTypeDefinition PP with details in [CostTypeDefinition PP](#).

### Set Cost Strategy Selection


Set selections in the CostSelection PP table which correspond with the current dependency level.

Table naming convention: <<dependency>>CostSelection.

Examples: AsiaCostSelection, GlobalCostSelection,...

### Using Cost Strategy Engine Parameter (Optional)

For each SKU in a PL/PG run, only its selected cost strategies will be loaded. To pass a cost strategy value as a parameter to a pricing engine, the cost strategy must be selected in the corresponding row in the CostTypeDefinition PP table. Then, add the parameter name in the desired pricing strategy, in the StrategyCalculationParameters property. For details, see [StrategyDefinition PP](#).

 The parameters are passed as inputs to engines and the input order is important, so do not change the default engine's parameters order.

## PSP Product Competition Module

This module gathers and displays product competition data. This can be used for any competition-based strategy.

### Module Related Elements

Technical Name	Label	Available in Independent PL/PG	Available in Dependent PL/PG	UI Visible	Output Type	Description
RawCompetitionData		Yes	Yes	No	List	The list of competition data obtained from the Competition Data master table
CompetitionData	Competition Data	Yes	Yes	Yes	Matrix	Formatted competition data
RawRelevantCompetitionData		Yes	Yes	No	List	The list of target competitors data obtained from the Competition Data master table
RelevantCompetitionData		Yes	Yes	Yes	Matrix	Formatted relevant competition data

	Relevant Competition Data					
--	---------------------------	--	--	--	--	--

#### Mechanism

	<i>Competition data</i>	<i>Relevant competition data</i>
Dependency mapping	Set the dependency mapping in the DependencyMapping PP table with the key <code>Product Competition</code> . Details can be found at <a href="#">DependencyMappingConfig PP</a> .	
Lookup	Search for records of the current SKU which are at the corresponding dependency level in the CompetitionData master table.	<ol style="list-style-type: none"> <li>Search for target competitors in the corresponding RelevantCompetitionData PP table. <ul style="list-style-type: none"> <li>Table name convention: &lt;&lt;dependency&gt;&gt;RelevantCompetitionData. Examples: RelevantCompetitionData, AsiaRelevantCompetitionData, GlobalRelevantCompetitionData,...</li> </ul> </li> <li>Filter the <code>Competition data</code> list to get only data from target competitors.</li> </ol>
Data fallback	If none is found, use the dependency fallback mechanism with the mapping type <code>Lookup</code> .	

#### Product Competition Module and Competition-based Strategies

If the module is turned off, null will be passed to the pricing engine. The resulting price will then be null as well and there will be a warning message indicating that there is no competition price to be calculated.

For more information about the competition pricing engine, see [Competition Engine](#).

## PSP Price Flexibility Module

This module provides integration with Price Flexibility Package (PFP). It adds the `Changes` element (technical name: `ChangesFromMonitor`) to the independent LPG which describes why a product has been automatically added to a price grid.

Currently, multiple features from PFP v1.2 are not supported. Price Flexibility Module will work properly only with configuration using a single monitor instance (or with older versions of PFP).

#### About Price Flexibility Package

The package is designed to notify a user group of the status when a product is added or updated. It means that changed products will be added to a Live Price Grid and wait for approval/denial by members of the assigned user group. Once the decision is made, approved products will be moved to a configured Price List and denied products will be removed from the Live Price Grid.

Price Flexibility Package monitors all products available in the system. Currently, it is not possible to filter them.

For more information about the Price Flexibility Package, see [Price Flexibility Overview](#).

# Price Setting Administration

This section contains more technical information useful for people who deploy, upgrade or extensively reconfigure the package.

- [How to Deploy Price Setting Accelerator](#)
- [How to Upgrade Price Setting Accelerator](#)
- [Modify Accelerator Price Setting Package](#)
- [Hot Swapping Capability](#)

## How to Deploy Price Setting Accelerator

- [Deployment Steps](#)
  - [Deploying Required Objects](#)
  - [Provide Dependency Configuration data](#)
  - [Set up Product Segmentation](#)
  - [Package Bootstrapping](#)
  - [Configuration and Price List/Grid Creation](#)

The only supported way to deploy Price Setting Package to a partition is via PlatformManager.

Access PlatformManager at <https://platform.pricefx.com/> and log in with your account or using O365.

Then follow the steps described in the PlatformManager documentation in [Package Deployment](#) to deploy "Price Setting Package" that you can find in the "Accelerator Packages" section.

### Deployment Steps

The process consists of a few steps; some require you to provide input data, some are more or less automated.

#### Deploying Required Objects

When you select a target partition and confirm the deployment, the first step is to deploy all the necessary Pricefx objects that build the package. Sit back and relax, as it may take a minute or two.

#### Provide Dependency Configuration data

Dependency Configuration is essential for this Accelerator. It defines what dependency levels (countries, channels, etc.) you will be using. You can find more information at [Dependent Price Lists and Data Fallbacks](#).

Examine your customer data and requirements before you fill in this configuration because any re-configuration will require you to go through some manual steps. You can find frequently needed procedures in <https://pricefx.atlassian.net/wiki/pages/resumedraft.action?draftId=2666823781>.

You will need to upload this configuration as a CSV file. We suggest to have attributelds in the first line of the file as headers, so that PlatformManager can automatically map it with the proper fields. Otherwise, manual mapping might be required. The file will be translated into [DependencyConfiguration PP](#), so you can import all the information that this PP accepts.

Sample file:



### Set up Product Segmentation

The next step will display a form which will let you configure product segmentation individually for each supported feature. More information can be found at [Product Segmentation](#).

Here it is also important to do a proper research before submitting the configuration because any re-configuration will require you to do some manual steps.

### Package Bootstrapping

The last step of package deployment is an automatically triggered bootstrapping. It will take all the information that you provided in previous steps and create and set up all the required tables. The background process is described in more details [here](#).

### Configuration and Price List/Grid Creation

Now that everything has been created you can start the configuration. You can go through the documentation and configure all required features manually by adjusting PPs, but we highly recommend using our "Price Setting Accelerator Configuration Wizard" to do it.

The last configuration task is to create Price Lists or Price Grids that you will use for your independent and dependent levels.

For an independent/standalone Price List/Grid, the selected logic should be "IndependentPriceListLogic". For others, "DependentPriceListLogic" should be used.

To use the [Volume Breakdown](#) functionality, you need to create a Price List/Grid of the MATRIX type, then in the Matrix logic select "VolumeBreakdownMatrixLogic" and in the Matrix logic element select "Volumes".

In the Price Lists/Grids, hide the Product Currency column (it gets the information from Product Master and it is not needed here it; currency is taken from the Currency column). To hide a column, use Preferences. You can also set default preferences for a price list in [Pricefx Configuration](#).

### How to Upgrade Price Setting Accelerator

This guide explains how to upgrade Accelerator to a new version.

Note: Optional features might need to be reconfigured after the upgrade. For details see the documentation of specific [modules/engines](#).

Sequence of steps:

- [Create Backup of All Logics](#)
- [Find Out Which Accelerator You Have](#)
- [Read All Manual Changes You Need to Perform](#)
- [Execute All Upgrade Notes](#)
- [Run "Price Setting Package - Upgrade"](#)
- [Execute Rest of Upgrade Notes](#)

### Create Backup of All Logics

Changes to logics will not be preserved through the upgrade. They will need to be applied again in the best case scenario. In the worst case scenario, they will need to be developed from scratch if the code has changed too much.

### Find Out Which Accelerator You Have

The easiest option to find out which Accelerator Package you have installed on your partition is to log in PlatformManager and navigate to Marketplace Templates Management Deployed Templates and search for your package and partition.

### Read All Manual Changes You Need to Perform

Read through this directory: [Price Setting Package Release Notes](#)

All manual changes will be marked as "Upgrade notes" or similarly.

 You need to check all versions between your version and the target one, not only newest/major one.

### Execute All Upgrade Notes

Follow all of the instructions in the release notes guiding you what to do before upgrading.

Remember that upgrade notes are created additively, so you should do them in the proper order.


### Run "Price Setting Package - Upgrade"

Go to PlatformManager Marketplace Accelerator Packages Price Setting Package - Upgrade and perform the upgrade.

### Execute Rest of Upgrade Notes

If there is no indicator when the upgrade note should be executed, it means it should be executed now.

### Modify Accelerator Price Setting Package

 We do not support custom modifications, so before raising any issue in Accelerator, please make sure it happens on a clean deployment without any of your modifications.

As Accelerators utilize the usual common Pricefx objects and wrapped pfxpackage tool for deployment, they are bound with the same restrictions as any other customer project. The biggest one is that upgrading the package overwrites objects existing on the partition. For this particular reason, any changes made to the package logics will be overwritten during an Accelerator upgrade and will have to be re-applied manually.

To make this Accelerator extendable by customer projects, we split our logics into tiers:

- **Tier One** - User facing logics. These define what data is shown to the user in price lists / grids together with some hidden technical elements. We try to implement these elements as simple "dispatchers", so usually the only thing that they do is to gather required inputs from previous elements and dispatch the calculation to Tier2 logic. Because it is the really easy to swap the dispatching method to some custom library, **this is where we recommend to make all of your modifications**. Upgrades should not cause any complex conflicts even if the underlying Accelerator business logic changes and re-applying modifications should be a breeze. Logics:
  - IndependentPriceListLogic
  - DependentPriceListLogic
  - VolumeBreakdownMatrixLogic
- **Tier Two** - Common logic libraries. This is where most Accelerators business requirements are implemented. It is full of detailed utilities that calculate and prepare data for Tier1 logics. This is where most package's complexity is handled. **Changes at this tier are still possible but not particularly encouraged**. Just be aware that resolving conflicts after upgrading this logic may be quite demanding. Logics:
  - PriceBuilderCommonElementsUtil
- **Tier Three** - Library logics. These are calculations and utilities that are designed to be working independently from the rest of the package. This tier stores for example all calculation engines' implementations. Over time this logic became smaller and smaller because we moved some of the existing utilities to the Pricefx Groovy Library (SharedLib). **Changes at this tier are not recommended**. Logics:
  - PriceListManagement
  - SharedLib from Shared Groovy Library
- **Tier X** - All remaining logics that do not directly affect business requirements. Changing them will usually not be necessary, but if for some reason it is - you can treat them as Tier2 when it comes to upgradeability. Logics:
  - PriceInsightsDashboardLogic
  - PriceSettingPackageInputConfigurator
  - PSP\_ConfigWizard
  - PSP\_ConfigWizardCommonLib
  - PSP\_ConfigWizardExecutor
  - PSP\_ConfigWizardScreenFactory

## Hot Swapping Capability

Hot swapping means changing configuration without damaging the feature itself.

Here is some general advice on how to handle configuration Price Parameters:

- All configurations work based on column names (UI name) and/or column labels/metadata (technical names). They do not work based on column *labels* (UI name) or labelTranslation (technical name).
- Do not change the name of any configuration PP column. It might be especially tempting in "DependencyConfiguration" PP, but don't do it.
- Column names of sample data can be freely changed assuming that the configuration will be properly adjusted.
- Values of dynamically created Price Parameters can be changed without redeployment. They act as standard Pricefx Price Parameters. The full list can be found [here](#). Values can be added/removed as needed. If the user wants to handle all products without fallback to another DependencyLevel, we recommend to have one value with an asterisk "\*" on every key. It provides a fallback without leaving the scope of the current Dependency Level configuration.

- Changing dynamically created Price Parameters requires caution, but can be done according to [this](#) guide.
- For PricesSettingLevel PP modification
- PriceSettingConfig and other configuration PPs should be changed with extreme caution. Technically every configuration can be changed during the system run but changes can affect a lot of things. That is why you should be careful here and customers are not even recommended to do this on their own in PROD environments without testing changes on some QA instance.
- Making changes to price setting levels is tricky. See [Adjustments after changing Price Setting Level](#).

## Price Setting Technical Information


This is the main section for documentation on PSP architecture. You can find here information about inner workings of the package and more advanced configuration concept descriptions.

- [Elements Documentation](#)
- [Price Setting Price Parameter Configuration](#)
  - [Standard Configuration](#)
  - [Bootstrapped Tables](#)
    - [Base Strategy Selection Lookup](#)
    - [Strategy Selection Lookup](#)
    - [Cost Selection Lookup](#)
    - [Minimum Margin Lookup](#)
    - [Dependency Level Adjustment Lookup](#)
    - [Discount Lookup](#)
    - [Additional Discount Lookup](#)
    - [Cost Plus Lookup](#)
    - [Price Increase Lookup](#)
    - [Adjusted Price Corridor Lookup](#)
    - [List Price Corridor Lookup](#)
    - [Relevant Competition Data Lookup](#)
    - [Volume Breakdown Lookup](#)
    - [Margin Alert Lookup](#)
- [Price Setting Config PP](#)
  - [Cost Lookup](#)
  - [Actual Price Lookup](#)
  - [Stock Lookup](#)
  - [Exceptions and Manual Override Allowance Config \(PriceSettingConfig\)](#)
  - [Exception Lookup](#)
  - [Rounding Rules Lookup](#)
  - [Transaction Lookup](#)
  - [Forecast Config \(PriceSettingConfig\)](#)
  - [Forecast Lookup](#)
  - [Last Period Config \(PriceSettingConfig\)](#)
- [Module Configuration Tables](#)
  - [PriceSettingModules PP](#)
  - [For PSP Advanced Cost Module](#)
    - [Advanced Cost Lookup \(CostTypeDefinition PP\)](#)
  - [For PSP Strategy Conditions Module](#)
    - [StrategyConditions PP](#)

- For PSP Override Module
    - PricingExceptions PP
- Other Configs
  - DependencyMappingConfig PP
  - DependencyConfiguration PP
  - ExchangeRates PP
  - PriceSettingDimensions PP
  - PriceSettingLevel PP
  - StrategyDefinition PP
  - VolumeBreakdownExceptions PP
  - WarningConfig PP
  - CompetitionAdditionalConfig PP
- Error Handling Deep Dive
- Caching Lookup Results
- Batching
- Price Setting Configuration Wizard - Technical Design

## Elements Documentation

All significant elements are already documented as groovyDocs: <https://gitlab.pricefx.eu/accelerators/price-builder-accelerator>

 Be aware of the Pricefx field called "Manual Override". It is an out-of-the-box field which is not related to Accelerators and should not be used.

## Price Setting Price Parameter Configuration

- Standard Configuration
- Bootstrapped Tables
- Price Setting Config PP
- Module Configuration Tables
- Other Configs

### Standard Configuration

The Price Setting Package comes with some standard configuration. This includes some structures (PX, PP) as well as pre-configured data in configuration PPs.

The following items are included in a standard configuration:

- Product Extensions
  - ProductCosts - Used by core Cost lookups
  - ListPrices - Used by Actual Price lookups
  - PromotionPrices - Used by sample engine configuration
  - RecommendedRetailPrices - Used by sample engine configuration
- Price Parameters
  - Sample configurations for various pre-configured engines
    - AnchorAdditionalConfig
    - AttributeBasedPricingRules
    - AvgCompetitionAdditionalConfig
    - CostPlusAdditionalConfig

- MaxCompetitionAdditionalConfig
- MinCompetitionAdditionalConfigMin
- PriceIncreaseAdditionalConfig
- PromotionLookupEngineConfig
- RRPLookupEngineConfig
- Data containers configured for different features:
  - AnchorData
  - PricingExceptions - Data container for strategy and price exceptions
  - StockData - Data container for product stock data
- Pre-Configuration for Price Setting Package that **has to be completed**:
  - Configuration for lookups in included data containers
  - Configuration for dependency mapping in some of the included data containers and product extensions
  - Sample of complete configuration that has to be completed
- Sample for Pricing Strategy Definition
- Sample for Rounding Rules Configuration
- Sample for Pricing Exceptions

### Bootstrapped Tables

- [Base Strategy Selection Lookup](#)
- [Strategy Selection Lookup](#)
- [Cost Selection Lookup](#)
- [Minimum Margin Lookup](#)
- [Dependency Level Adjustment Lookup](#)
- [Discount Lookup](#)
- [Additional Discount Lookup](#)
- [Cost Plus Lookup](#)
- [Price Increase Lookup](#)
- [Adjusted Price Corridor Lookup](#)
- [List Price Corridor Lookup](#)
- [Relevant Competition Data Lookup](#)
- [Volume Breakdown Lookup](#)
- [Margin Alert Lookup](#)

### Base Strategy Selection Lookup

*Name*

{nameOfDependency}BaseStrategySelection

*Attributes*

- Price Strategy #1
- Price Strategy #2
- Price Strategy #3
- Price Strategy #4
- Price Strategy #5

*Description*

Order of used base strategies. Base strategies appear before the standard strategies. However, if they fail, they are removed from "Prices" popup.

## Strategy Selection Lookup

### *Name*

{nameOfDependency}StrategySelection

### *Attributes*

- Price Strategy #1
- Price Strategy #2
- Price Strategy #3
- Price Strategy #4
- Price Strategy #5
- Prioritize Independent Level Price

### *Description*

Order of used strategies. Other strategies might appear before, if exceptions/overrides/base strategies are used. "Prioritize Independent Level Price" is relevant only for Dependent PL/PG - it defines if a price from master Dependency Level should be placed at the first place or at the end.

## Cost Selection Lookup

### *Name*

- {nameOfDependency}CostSelection

### *Attributes*

- Cost Type #1
- Cost Type #2
- Cost Type #3
- Cost Type #4
- Cost Type #5

### *Description*

Order of Cost Types calculated for the Advanced Cost module.

## Minimum Margin Lookup

### *Name*

{nameOfDependency}MinMargin

### *Attributes*

- Min Margin %

### *Description*

Value of the minimum margin in percent for the Price Checks module.

## Dependency Level Adjustment Lookup

### *Name*

- {nameOfDependency}DependencyLevelAdjustment

*Attributes*

- Adjustment %

*Description*

Valid only for Dependent PL/PG. Value of Price Adjustment between master Dependency Level Price and current Dependency Level.

**Discount Lookup**

*Name*

{nameOfDependency}Discount

*Attributes*

- Discount %

*Description*

Value of minimum Discount in percent for Net Price module. This value is a difference between Gross and Net (Final) Price.

**Additional Discount Lookup**

*Name*

{nameOfDependency}AdditionalDiscount"DependencyConfiguration"

*Attributes*

- Discount %

*Description*

Value of minimum Discount in percent for the Target Price engine. It is one of the supportive parameters coming with pre-configured strategies. This value is a difference between Final Price (at the end of calculation) and selling price (e.g. rebates).

**Cost Plus Lookup**

*Name*

{nameOfDependency}CostPlus

*Attributes*

- Plus %
- Plus Absolute

*Description*

It is one of the supportive parameters coming with pre-configured strategies. Depending on additional configuration of Cost+ strategy, Plus % OR Plus Absolute will be used.

**Price Increase Lookup**

*Name*

## {nameOfDependency}PriceIncrease

### *Attributes*

- Plus %
- Plus Absolute

### *Description*

It is one of the supportive parameters coming with pre-configured strategies. Depending on additional configuration of PriceIncrease strategy, Plus % OR Plus Absolute will be used.

## Adjusted Price Corridor Lookup

### *Name*

- AdjustedPriceCorridor

### *Attributes*

- Minimum Absolute
- Minimum Corridor
- Maximum Corridor
- Maximum Absolute

### *Description*

Value of Adjusted Price Corridor thresholds in percent for the Price Checks module. Difference between the price from Master Dependency Level and current price is compared and quotient is showed as an alert, colored depending on this config.

## List Price Corridor Lookup

### *Name*

- ListPriceCorridor

### *Attributes*

- Minimum Absolute
- Minimum Corridor
- Maximum Corridor
- Maximum Absolute

### *Description*

Value of List Price Corridor thresholds in percent for the Price Checks module. Difference between the price from Master Dependency Level and current price is compared and quotient is showed as an alert, colored depending on this config.

## Relevant Competition Data Lookup

### *Name*

- {nameOfDependency}RelevantCompetitionData

### *Attributes*

- Competitor #01-#29

#### *Description*

List of relevant competitors. It filters out values from [Competition Data Lookup](#) to create a list of Relevant Competitors, used for a popup display and pricing.

### Volume Breakdown Lookup

#### *Name*

- {nameOfDependency}VolumeBreakdown

#### *Attributes*

- Volume #01
- Discount #01
- ...
- Volume #15
- Discount #15

#### *Description*

Pairs consisting of Volumes in integers and Volume Discounts in percent. These values will be used in generating Matrix PG and discounts will be applied to prices.

### Margin Alert Lookup

#### *Name*

- MarginAlertsForPriceLists

#### *Attributes*

- Critical Threshold
- High Threshold
- Medium Threshold

#### *Description*

It is the value of Margin Alerts thresholds in percent for the Price Checks module. The `Margin Flag` element in the PL/PG has the values Critical, High, or Medium, depending on what margin falls into what range. For Critical and High ranges, it will color the row in red. For the Medium range, it will color the row in yellow.

The checking order is: critical threshold => high threshold => medium threshold (too small margin => small margin => medium margin). It is not in data order (not from the smallest to biggest or vice versa).

**Example:** Given medium threshold = 45, high threshold = 100, critical threshold = 20, margin = 38.89, the checking process will be:

- $38.89 \leq \text{critical (20)}$ ? No
- $38.89 \leq \text{high (100)}$ ? Yes

Therefore, the margin falls into the high range.

Company Parameter Values: Margin Alerts For Price Lists

Business Unit: Search... Product Group: Search... Product Class: Search... Medium Threshold: Search... High Threshold: Search... Critical Threshold: Search...

\* \* \* \* 45.00% \* \* \* \* 20.00%

← 1559 (Germany\_2112)

les Volume YTD	List Price Corridor	Adjusted Price Corri...	Revenue Breakeven ...	Margin Breakeven Vo...	Minimum Margin	Margin Flag	New Margin
6,400	28.22%	42.45%	△	△	1.11%	Critical	20.00%
	163.19%	△	△	△	1.11%	Medium	38.89%
	100.00%	122.22%	△	△	1.11%		100.00%

Company Parameter Values : Margin Alerts For Price Lists [1]

Business Unit: Search... Product Group: Search... Product Class: Search... Medium Threshold: Search... High Threshold: Search... Critical Threshold: Search...

\* \* \* \* 45.00 % \* \* \* \* 100.00 % \* \* \* \* 20.00 %

← 1559 (Germany\_2112)

les Volume YTD	List Price Corridor	Adjusted Price Corri...	Revenue Breakeven ...	Margin Breakeven Vo...	Minimum Margin	Margin Flag	New Margin
6,400	28.22%	42.45%	△	△	1.11%	Critical	20.00%
	163.19%	△	△	△	1.11%	High	38.89%
	100.00%	122.22%	△	△	1.11%	High	100.00%

## Price Setting Config PP

Each of the subpages represents a single row in the PriceSettingConfig Price Parameter.

- [Cost Lookup](#)
- [Actual Price Lookup](#)
- [Stock Lookup](#)
- [Exceptions and Manual Override Allowance Config \(PriceSettingConfig\)](#)
- [Exception Lookup](#)
- [Rounding Rules Lookup](#)
- [Transaction Lookup](#)
- [Forecast Config \(PriceSettingConfig\)](#)
- [Forecast Lookup](#)
- [Last Period Config \(PriceSettingConfig\)](#)

### Cost Lookup

This configuration represents a single row of the PriceSettingConfig Price Parameter.

This configuration row can be set up separately for independent and dependent price lists. It means that there can be maximum 3 rows with the key "Cost", each with a different value in the "Condition" column.

Column	Value	Description
Key	Cost	Holds the product cost configuration.

Con diti on	*	Condition is not needed here. Leave it at the default value.
Type	Lookup	
Sou rce	PX	Currently, only a PX lookup is supported.
Sou rce Table	{PX name}	Remember to use "name" from the configuration, not a label.
Sou rce Field	{name of the column with the product cost}	
Sou rce Fiel d 2	Currency	If there is a currency in the PX Cost table, get it and do the conversion. If there is none, then get the LPG/PG BaseCurrency.
Sou rce Fiel d 3	{name of the column with the start date of the validity period, typically ValidFrom}	Used for searching with a validity period. <ul style="list-style-type: none"> <li>• When the ValidFrom field (Source Field 2) is left empty, the ValidTo field (Source Field 3) will be ignored and the normal search approach will be applied.</li> <li>• When the ValidFrom field is set but the ValidTo field is left empty, it will find the value with the closest validity date.</li> <li>• When both ValidFrom and ValidTo fields are set, it will find the value falls in the time range between ValidFrom source value and ValidTo source value.</li> </ul>
Sou rce Fiel d 4	{name of the column with the end date of the validity period, typically ValidTo}	

### Actual Price Lookup

This configuration represents a single row of the PriceSettingConfig Price Parameter.

Column	Value	Description
Key	Actual Price	Defines how to lookup values for the ProductListPrice lookup.
Condition	*	Condition is not needed here. Leave it at the default value.
Type	Lookup	Hardcoded value.
Source	<ul style="list-style-type: none"> <li>• PX</li> <li>• PL</li> <li>• PG</li> </ul>	Source type for the lookup. PG might be used only when using Price Grids. It will not work for Price List.

Source Table	{name of the PX}	Needed only when we use PX as a source. For PL it is always the last approved price for a given dependency level name.
Source Field	{name of the column with price information in the source data}	Needed only when we use PX as a source.
Source Field 2	{name of the column with the start date of the validity period}	Needed only when we use PX as a source.
Source Field 3	{name of the column with the end date of the validity period}	Needed only when we use PX as a source.
Source Field 4	{name of the column with the currency}	Needed only when we use PX as a source.

#### Why This Lookup Is Not Standardized


Technically, Actual Price PX lookup is standardized. However, from the configuration point of view, flow of data reading is non-standard when using PL and PG sources.

PL and PG do not use batching. PG is not even a lookup as such, since it reads data from the previous run of the same price grid.

#### Stock Lookup

This configuration represents a single row of the PriceSettingConfig Price Parameter.

This configuration row can be set up separately for independent and dependent price lists. It means that there can be maximum 3 rows with the key "Stock", each with a different value in the "Condition" column.

Column	Value	Description
Key	Stock	Holds the product stock configuration.
Condition	*	Condition is not needed here. Leave it at the default value.
Type	Lookup	
Source	<ul style="list-style-type: none"> <li>PX</li> <li>PP</li> </ul>	Type of the source table.
Source Table	{name of PX or PP}	Remember to use "name" from the configuration, not a label.  For PP lookups, we assume two things: <ol style="list-style-type: none"> <li>When dependency mapping is not applied, there is only one key which is the SKU code</li> <li>For other cases: <ol style="list-style-type: none"> <li>key1 - SKU code</li> </ol> </li> </ol>

		b. key2 - Dependency mapping value c. (optional) key3 - Valid from d. (optional) key4 - Valid to
Source Field	{name of the column with the stock data for a product}	
Source Field 2	{name of the column with the start date of the validity period, typically ValidFrom}	Used for searching with a validity period. <ul style="list-style-type: none"> <li>When the ValidFrom field (Source Field 2) is left empty, the ValidTo field (Source Field 3) will be ignored and the normal search approach will be applied.</li> <li>When the ValidFrom field is set but the ValidTo field is left empty, it will find the value with the closest validity date.</li> <li>When both ValidFrom and ValidTo fields are set, it will find the value which falls in the time range between the ValidFrom source value and ValidTo source value.</li> </ul>
Source Field 3	{name of the column with the end date of the validity period, typically ValidTo}	

### Exceptions and Manual Override Allowance Config (PriceSettingConfig)

This configuration represents a single row of the PriceSettingConfig Price Parameter.


One configuration row should exist for every key-condition pair mentioned in the table below.

Column	Value	Description
Key	<ul style="list-style-type: none"> <li>Independent Manual Override Allowance</li> <li>Dependent Manual Override Allowance</li> </ul>	Holds the configuration of exception/override allowance. One row should be configured for every entry.
Condition	<ul style="list-style-type: none"> <li>Strategy</li> <li>Price</li> </ul>	Defines which exception type is configured. There is one row for each condition.
Type	<ul style="list-style-type: none"> <li>Yes</li> <li>No</li> <li>LineLevel</li> <li>ExceptionTable</li> </ul>	Defines if and what kind of exception is allowed: <ul style="list-style-type: none"> <li>LineLevel - Only line level manual overrides are available. Users can define them within PL/PG per product. Exception table records are not checked at all.</li> <li>ExceptionTable - Only exceptions through the exception tables are available (defined in <a href="#">ExceptionConfig</a>). It also means that line level manual overrides in elements "ManualPrice", "ManualPriceReason" and "PriceSelector" are cleared.</li> <li>Yes - Both exception types are allowed. The importance hierarchy is as follows:</li> </ul>


		<ol style="list-style-type: none"> <li>1. Price Manual Override</li> <li>2. Strategy Manual Override</li> <li>3. Price Exception</li> <li>4. Strategy Exception</li> </ol> <ul style="list-style-type: none"> <li>• No - Exceptions are not allowed. Elements "ManualPrice", "ManualPriceReason" and "PriceSelector" are hidden from the user.</li> </ul>
--	--	---

### Exception Lookup

This configuration describes two rows of the PriceSettingConfig Price Parameter.

 Exception Lookup will not be performed, if allowance config forbids it: [Exceptions and Manual Override Allowance Config \(PriceSettingConfig\)](#).

One configuration row should exist for every key-condition pair mentioned in the table below.

Column	Value	Description
Key	<ul style="list-style-type: none"> <li>• Strategy Exception</li> <li>• Price Exception</li> </ul>	Holds the exception configuration. There is one row for each type.
Condition	*	Condition is not needed here. Leave it at the default value.
Type	Lookup	
Source	<ul style="list-style-type: none"> <li>• PX</li> <li>• PP</li> </ul>	Type of the source table.
Source Table	{name of PX or PP}	<p>Remember to use "name" from the configuration, not a label.</p> <p> For PP lookups, we assume two things:</p> <ol style="list-style-type: none"> <li>1. The only existing key is the SKU code, dependency mapping is not applied.</li> <li>2. If there are two keys: <ol style="list-style-type: none"> <li>a. key1 - SKU code</li> <li>b. key2 - Dependency mapping value</li> </ol> </li> </ol>

b le		
S o u r c e F i e l d	{name of the column with the exception data for a product}	<p>The type of this column depends on the exception type.</p> <ul style="list-style-type: none"> <li>• Strategy - String with the name of an existing strategy definition. <b>Important note:</b> Strategy exception can be selected only from strategies configured in the StrategySelection PP for a given product. It means that if we try to use a strategy that is not calculated by default for the product, this exception will be ignored.</li> <li>• Price - Numeric value with a price override.</li> </ul>
S o u r c e F i e l d 2	{name of the column with the start date of the validity period, typically ValidFrom}	<p>Used for searching with a validity period.</p> <ul style="list-style-type: none"> <li>• When the ValidFrom field (Source Field 2) is left empty, the ValidTo field (Source Field 3) will be ignored and the normal search approach will be applied.</li> <li>• When the ValidFrom field is set but the ValidTo field is left empty, it will find the value with the closest validity date.</li> <li>• When both ValidFrom and ValidTo fields are set, it will find the value which falls in the time range between the ValidFrom source value and ValidTo source value.</li> </ul>
S o u r c e F i e l d 3	{name of the column with the end date of the validity period, typically ValidTo}	
S o u r c e F i e l d 4	{name of the column with currency}	Configuration for optional currency conversion. No data means no conversion.

## Rounding Rules Lookup

This configuration represents a single row of the PriceSettingConfig Price Parameter.

Column	Value	Description
Key	Rounding Rules	Defines how to look up values for the Rounding Rules lookup.
Condition	*	Condition is not needed here. Leave it at the default value.
Type	Lookup	Hardcoded value.
Source	PP	Hardcoded value.
Source Table	{name of the PP}	Name of the PP table where rules are stored
Source Field	{name of the column with Rounding Rule}	Remember to use "name" from the configuration, not a label.
Source Field 2	{name of the column with Rounding Mode}	Remember to use "name" from the configuration, not a label.
Source Field 3	{name of the column with the start date of the validity period, typically ValidFrom}	Used for searching with a validity period. <ul style="list-style-type: none"> <li>When the ValidFrom field is left empty, the ValidTo field will be ignored and the normal search approach will be applied.</li> <li>When the ValidFrom field is set but the ValidTo field is left empty, it will find the value with the closest validity date.</li> <li>When both ValidFrom and ValidTo fields are set, it will find the value which falls in the time range between ValidFrom and ValidTo source values.</li> </ul>
Source Field 4	{name of the column with the end date of the validity period, typically ValidTo}	

### Why This Lookup Is Not Standardized

We do not perform lookup per SKU.

## Transaction Lookup

This configuration represents a single row of the PriceSettingConfig Price Parameter.

Column	Value	Description
Key	Transaction Source	Defines where data about transactions are stored.
Condition	*	Condition is not needed here. Leave it at the default value.
Type		Leave blank.

Source	<ul style="list-style-type: none"> <li>• Datamart</li> <li>• Datasource</li> <li>• PX</li> </ul>	Type of the table where transactions are stored.
Source Table	{name of the table with transactions}	
Source Field	{name of the column with the invoice price}	Used for turnover calculation.
Source Field 2	{name of the column with the quantity field}	Used for volume calculation.
Source Field 3	{name of the column with SKU}	Only PX has a default SKU, so we require SKUs for all tables for consistency.
Source Field 4	{name of the column with the date for the invoice}	⚠ The column must be of the Date type.
Source Field 5	{name of the column with the currency}	Invoice price currency
Source Field 6	{name of the column with minimum value of aggregated data}	Only for pre-aggregated PX Source
Source Field 7	{name of the column with maximum value of aggregated data}	Only for pre-aggregated PX Source
Source Field 8	{name of the column with average value of aggregated data}	Only for pre-aggregated PX Source

#### Why This Lookup Is Not Standardized

- Hierarchy works differently for Transaction data (it goes down for all sub-levels of the dependency, instead of working as a fallback).
- A lot of data will be returned for the given time period, there is a "date overlap" issue.
- Lookup manager supports api.stream calls, but transaction data might be stored in a Datamart or Data Source.

#### Forecast Config (PriceSettingConfig)

This configuration represents a single row of the PriceSettingConfig Price Parameter.

There are rows for forecasts, one for every quarter, so there will be key pairs "Forecast-Q1", "Forecast-Q2" etc.

Each quarter can be configured with one of these three types:

- [Last Year](#)
- [Linear](#)
- [Lookup](#)

#### Last Year

Column	Value	Description
--------	-------	-------------

Key	Forecast	Defines how to calculate the forecast.
Condition	<ul style="list-style-type: none"> <li>• Q1</li> <li>• Q2</li> <li>• Q3</li> <li>• Q4</li> </ul>	Logic checks the current date. If it matches this configuration row, the row is used.
Type	LastYear	The forecast will be equal to last year's sales (based on the transaction source).

#### Linear

Column	Value	Description
Key	Forecast	Defines how to calculate the forecast.
Condition	<ul style="list-style-type: none"> <li>• Q1</li> <li>• Q2</li> <li>• Q3</li> <li>• Q4</li> </ul>	Logic checks the current date. If it matches this configuration row, the row is used.
Type	Linear	The forecast will be calculated with a linear growth based on the year to date values from the transaction source.

#### Lookup


See [Forecast Lookup](#).

#### Forecast Lookup

This configuration represents a single row of the PriceSettingConfig Price Parameter.

⚠ There are also other ways of reading Forecast; Lookup is only one of them.

Column	Value	Description
Key	Forecast	Defines how to calculate the forecast.
Condition	<ul style="list-style-type: none"> <li>• Q1</li> <li>• Q2</li> <li>• Q3</li> <li>• Q4</li> </ul>	If the current date matches this configuration row, the row is used. This is checked by the logic.
Type	Lookup	The logic will perform a lookup for the forecast from an external source.
Source	<ul style="list-style-type: none"> <li>• Datamart</li> <li>• Datasource</li> <li>• PX</li> </ul>	Source type where the forecasts are stored.
Source Table	{name of the table with forecasts}	

Source Field	{name of the column with the turnover}	
Source Field 2	{name of the column with the quantity}	
Source Field 3	{name of the column with SKU}	Only PX has a default SKU, so we require SKUs for all tables for consistency.
Source Field 4	{name of the column with the invoice date}	The forecast will always be fetched using only data with the "next year" filter applied on this column.  The column must be of the Date type.
Source Field 5	{name of the column with currency of turnover}	Turnover currency
Source Field 6	{name of the column with minimum value of aggregated data}	Only for pre-aggregated PX Source
Source Field 7	{name of the column with maximum value of aggregated data}	Only for pre-aggregated PX Source
Source Field 8	{name of the column with average value of aggregated data}	Only for pre-aggregated PX Source

#### Why This Lookup Is Not Standardized

- Hierarchy works differently for Transaction data (it goes down for all sub-levels of dependency, instead of working as fallback).
- A lot of data will be returned for a given time period, there is a "date overlap" issue.
- Lookup manager supports api.stream calls, while transaction data might be stored in Datamart or Data Source.

#### Last Period Config (PriceSettingConfig)


This configuration represents a single row of the PriceSettingConfig Price Parameter. It is not related to lookup sources. It is a candidate to be moved to the module-specific table. For now, ignore column names in that config.

Column	Value	Description
Key	Last Period Transaction	Defines from which period data should be looked up
Codition	*	Condition is not needed here. Leave it at the default value.
attribute1	<ul style="list-style-type: none"> <li>• Years</li> <li>• Months</li> <li>• Weeks</li> <li>• Days</li> </ul>	Time unit of lookup configuration
attribute2	{any integer}	Amount of time units of lookup configuration

## Module Configuration Tables

- [PriceSettingModules PP](#)
- [For PSP Advanced Cost Module](#)
- [For PSP Strategy Conditions Module](#)
- [For PSP Override Module](#)

### PriceSettingModules PP

Column Name	Name	Status
Values	<ul style="list-style-type: none"> <li>• PSP_ADVANCED_COST</li> <li>• PSP_NET_PRICE_MODULE</li> <li>• PSP_OVERRIDES_MODULE</li> <li>• PSP_PRICE_CHECKS_MODULE</li> <li>• PSP_PRICE_FLEXIBILITY_MODULE</li> <li>• PSP_PRODUCT_COMPETITION_MODULE</li> <li>• PSP_ROUNDING_RULES_MODULE</li> <li>• PSP_STRATEGY_CONDITION_MODULE</li> <li>• PSP_TRANSACTION_MODULE</li> </ul>	<ul style="list-style-type: none"> <li>• On</li> <li>• Off</li> </ul>
Description	<p>Technical name of the module.</p> <p> All values needs to be present in the PP for the package to work correctly.</p>	Defines whether any given module is enabled or disabled.

Example:

## Price Parameter Values : PriceSettingModules [9]

<input type="checkbox"/>	Name	Status
<input type="checkbox"/>	PSP_PRODUCT_COMPETITION_MODULE	On
<input type="checkbox"/>	PSP_TRANSACTION_MODULE	On
<input type="checkbox"/>	PSP_NET_PRICE_MODULE	On
<input type="checkbox"/>	PSP_OVERRIDES_MODULE	On
<input type="checkbox"/>	PSP_PRICE_CHECKS_MODULE	On
<input type="checkbox"/>	PSP_PRICE_FLEXIBILITY_MODULE	On
<input type="checkbox"/>	PSP_STRATEGY_CONDITION_MODULE	On
<input type="checkbox"/>	PSP_ROUNDING_RULES_MODULE	Off
<input type="checkbox"/>	PSP_ADVANCED_COST	Off

### For PSP Advanced Cost Module

- [Advanced Cost Lookup \(CostTypeDefinition PP\)](#)

Advanced Cost Lookup (CostTypeDefinition PP)

Column name	Cost Type	Calculation Engine Suffix	Type	Calculation Method	Source Table	Source Field	Valid From	Valid To	Dependency Field	Dependency Type	Mapping Source Field	Currency
Values	Any string	Any string	"Lookup"	<ul style="list-style-type: none"> <li>SINGLE</li> <li>AVG</li> <li>SUM</li> </ul>	Name of the PX	Name of the PX column with cost	Name of the PX column with a valid From date (optional)	Name of the PX column with a valid To date (optional)	Name of the column in <a href="#">DependencyConfiguration</a> (country mapping feature)	<ul style="list-style-type: none"> <li>Table</li> <li>Lookup (country mapping feature)</li> </ul>	Name of the PX column (country mapping feature)	Name of the PX column with currency code
Description	Name of the definition, used in cost selection.	String concatenated with "COST" and passed to engines as a parameter.	Other types will be added in the future.	For details see <a href="#">Advanced Cost</a> .	Note: Name is required here, not a label.							

### For PSP Strategy Conditions Module

- [StrategyConditions PP](#)

StrategyConditions PP

For additional details see also [PSP Strategy Conditions Module](#).

Column name	Order	Condition	Rule	CheckException
Values	{incrementing integer}	{condition}	{rule}	<ul style="list-style-type: none"> <li>Yes</li> <li>No</li> </ul>
Description	Defines the order of conditions. Conditions are checked one by one and in case of multiple conditions and strategies, the result may vary based on the order.	Expression to be evaluated (boolean value to activate the rule). For details see <a href="#">PSP Strategy Conditions Module</a> .	Defines what to do with the strategy when the condition is evaluated to true. For details see <a href="#">Order</a> .	Determines if this condition should be applied to exception prices (both strategy and price exceptions). The check is performed only for strategies on the left side of the condition field.

### For PSP Override Module

- [PricingExceptions PP](#)

PricingExceptions PP

SKU	Dependent Level Name	Price Exception	Strategy Exception
-----	----------------------	-----------------	--------------------

<b>Column name</b>				
<b>Values</b>	{SKU from Product master}	{name of dependency level} as stated in <a href="#">DependencyConfiguration PP</a>	Defined price exception for given parameters	Defined strategy (as stated in <a href="#">StrategyDefinition PP</a> ) exception for given parameters
<b>Description</b>	e.g. "MB-0001"	e.g. "Global"	e.g. "20.0"	e.g. "Cost+"

Example:

Price Parameter Values : Pricing Exceptions [1]				
<input type="checkbox"/>	SKU	▼ Dependency Level Name	Price Exception	Strategy Exception
<input type="checkbox"/>	MB-0001	Global	20.00	

### Other Configs

The configuration options described here may be required to set up to enable individual Price Setting modules and features.

- [DependencyMappingConfig PP](#)
- [DependencyConfiguration PP](#)
- [ExchangeRates PP](#)
- [PriceSettingDimensions PP](#)
- [PriceSettingLevel PP](#)
- [StrategyDefinition PP](#)
- [VolumeBreakdownExceptions PP](#)
- [WarningConfig PP](#)
- [CompetitionAdditionalConfig PP](#)

### DependencyMappingConfig PP

Dependency mapping defines how different lookup data will be filtered when loaded from a dependent price list. It is available in the [DependencyMappingConfig PP](#) and it directly impacts configuration found in [PriceSettingConfig PP](#). Entries in this table have to be present even if you intend to use only the independent level price list.

Column	Value	Description
Key	<ul style="list-style-type: none"> <li>• Cost</li> <li>• Discount Group</li> </ul>	Defines how to filter out values from different dependencies. There must be one row for every key defined in this table.

	<ul style="list-style-type: none"> <li>• Transaction</li> <li>• Projection</li> <li>• Price Exception</li> <li>• Strategy Exception</li> <li>• Actual Price</li> <li>• Product Competition</li> <li>• Rounding</li> <li>• Stock</li> </ul>	
<p>Dependency Configuration PP}</p> <p>Field</p>		<p>The logic will lookup the Dependency Configuration PP for a dependency being calculated and take a value from the column configured in this field. It will be used as a filter for source data defined in the Source Table row. This filter will be applied to a column configured in Source Field of this configuration row. In the start, it has the "ADJUSTME" value and it must be changed.</p> <p>Special use cases:</p> <ul style="list-style-type: none"> <li>• Price Exception / Strategy Exception Dependency Mapping - We allow to use MATRIX typed PPs with only one key when the configured type is "PP". In this case, the Condition field value has to be "-" and dependency mapping will not be applied to these results.</li> </ul>
<p>Type</p>	<ul style="list-style-type: none"> <li>• Lookup</li> <li>• Table</li> </ul>	<p>We can do the dependency mapping mechanism in two ways:</p> <ul style="list-style-type: none"> <li>• <b>Lookup</b> - Assumes that you have a data source for all countries. e.g. one PX for the product cost for different products and values are defined by a label of the Mapping Source field. So the relationship is defined within the data source.</li> </ul>

	<ul style="list-style-type: none"> <li>• None</li> </ul>	<ul style="list-style-type: none"> <li>• <b>Table</b> - Assumes that you have multiple data sources for all dependencies. Each dependency has its own data source for a specified dependency mapping mechanism. When you use this type of search, you should also change the value of sourceTable in PriceSettingConfig: the value should include a placeholder that will be swapped with our dependency property defined by the dependency mapping mechanism. The placeholder has this format: &lt;&lt;DependencyPreference&gt;&gt;. In this type of value, the Mapping Source field is ignored because you do not need to filter results inside the data source. Table dependency does not work for competitors because Pricefx has only 1 PCOMP table. An example of usage: <ul style="list-style-type: none"> <li>• Dependency Level Name: Germany</li> <li>• Preference1 (from Dependency Configuration): DE</li> <li>• Dependency Field: Preference1</li> <li>• Source Type: PX</li> <li>• Source Table (from PriceSettingConfig): Product Costs &lt;&lt;DependencyPreference&gt;&gt; With this data, the dependency mapping mechanism will search for a PX named Product Costs DE and then perform the lookup.</li> </ul> </li> <li>• <b>None</b> - To use a Data Source with no dependency info.</li> </ul>
Mapping Source information in the source data	{name of the column with the dependency information in the source data}	<p>The label should be taken from the table defined in the lookup on the row from Source Table with the "Lookup" type. It filters results this way:  Filter.equal("\${Source Field}", "\${valueRetrievedFromCondition}")</p> <p>It is important to double-check if this field was set up correctly, as most data lookups in the package utilize api.stream() calls and they will return nothing if you request a field that does not exist on a target object type. For example - "Country" for type PCOMP will not return anything even though a given product has competition data, because the name of the field is "country".</p>

### DependencyConfiguration PP

Column Name	Dependency Level Name	Dependencies	Source Type	Source ID	Dimension	Currency	IsComplete	Preference #01-27
Values of	{name of	{independen	{type of source data	{ID of the source data for	Data used to help describe what the dependency is used for. If one level depends on another,	{	Flag indicating if all data is supposed to be filled at this point. If the dependency level contains "Yes", fallback	User defined data. These values are used in mapping the dependency level to the appropriate rows of a given source table. The mapping is managed via the DependencyMappingConfig PP

	de pe n d e n c y}	t l e v e l n a m e}	for i n d e p e n d e n t l o o k u p}	i n d e p e n d e n t l o o k u p}	that relation will be treated as HQ mode.	e n c y c o d e}	hierarchy will be stopped at this point.	table. Currently, only one field can be used to map at a time.
<b>D e s c r i p t i o n</b>	e. g. "G e r m a n y"	e. g. "G l o b a l"	Allowed values: <ul style="list-style-type: none"><li>• PG</li><li>• X PG</li><li>• PL</li><li>• X PL</li></ul>	e.g. "155"	e.g. "Country", "Warehouse"	e. g. "E U R"	Allowed values: <ul style="list-style-type: none"><li>■ Yes</li><li>■ No</li></ul>	e.g. "ISO Code", "SalesOrg"

Example:

**Price Parameter Values : DependencyConfiguration [8]**

<input type="checkbox"/>	Dependency ...	Depends On	Source Type	Source ID	Dimension	Currency	Is complete	ISO Code	SalesOrg
<input type="checkbox"/>	EU	Independent	*	*	Area	EUR	Yes	EU	SO00
<input type="checkbox"/>	Germany	EU	PG	155	Country	EUR	No	DE	SO20
<input type="checkbox"/>	Poland	Germany	PG	381	Country	PLN	No	PL	SO21
<input type="checkbox"/>	Asia	Independent	*	*	Area	EUR	Yes	GL	SO10
<input type="checkbox"/>	Warehouse1	Asia	*	*	Warehouse	EUR	No	UK	SO15
<input type="checkbox"/>	Warehouse2	Asia	*	*	Warehouse	EUR	No	UK	SO15
<input checked="" type="checkbox"/>	Webshop	Germany	PG	381	Channel	EUR	No	DE	SO20
<input type="checkbox"/>	StationaryShop	Germany	PG	381	Channel	EUR	No	DE	SO20

### ExchangeRates PP

Col um n n a m e	From	To	ValidDate	Rate
<b>Val ues</b>	{code of the base currency matching the <a href="#">DependencyConfiguration</a> data}	{code of the target currency matching the <a href="#">DependencyConfiguration</a> data}	{from which date the currency is valid}	{ex c h a n g e r a t e v a l u e}
<b>Des c r i p t i o n</b>	e.g. "PLN"	e.g. "EUR"	In case of multiple entries with the same currency, the first entry after the valid date is chosen.	e.g. "5.05"

Example:

### Price Parameter Values : ExchangeRates [5]

From	To	ValidDate	Rate
VIR	EUR	28/08/2019	215
VIR	PLN	13/08/2019	84.15
VIR	PLN	27/08/2019	86
PLN	EUR	28/08/2019	4.1
EUR	PLN	28/08/2019	0.25

### PriceSettingDimensions PP

Column name	Dimension	Order	Feature Name	Field Name
Values	Products	<ul style="list-style-type: none"> <li>• 1</li> <li>• 2</li> <li>• 3</li> <li>• 4</li> <li>• 5</li> </ul>	<ul style="list-style-type: none"> <li>▪ Fallback</li> <li>▪ StrategySelection</li> <li>▪ BaseStrategySelection</li> <li>▪ MinMargin</li> <li>▪ CostPlus</li> <li>▪ PriceIncrease</li> <li>▪ AdditionalDiscount</li> <li>▪ RelevantCompetitors</li> <li>▪ DependencyAdjustment</li> <li>▪ VolumeBreakdown</li> <li>▪ AdjustedPriceCorridor</li> <li>▪ ListPriceCorridor</li> <li>▪ CostSelection</li> <li>▪ Discount</li> </ul>	{name of the column from the products table}
Description	Currently, "Products" is the only allowed value.	Hierarchy order where "1" is the most important attribute, the following ones are used for more specific lookups.	Name of the affected feature for which PP will be generated with given keys. Only "Fallback" is mandatory, features with no specified keys will use "Fallback".	

Example:

### Price Parameter Values : PriceSettingDimensions [7]

Dimension	Order	Key3	Field Name
Products	1	Fallback	Business Unit
Products	2	Fallback	Product Group
Products	3	Fallback	Product Class
Products	1	StrategySelection	Business Unit
Products	2	StrategySelection	Product Group
Products	1	BaseStrategySelection	Product Class
Products	2	BaseStrategySelection	Business Unit

### PriceSettingLevel PP

 See also [Adjustments after Changing Price Setting Level.](#)

<b>Column name</b>	<b>Pricelist</b>	<b>Price level</b>
<b>Values</b>	<ul style="list-style-type: none"> <li>• {name of dependency level } as stated in <a href="#">DependencyConfiguration</a> PP</li> </ul>	<ul style="list-style-type: none"> <li>• Gross</li> <li>• Gross / Net</li> </ul>
<b>Description</b>		If an additional discount from a gross to net price should be calculated for a specific dependency level.


Example:

Price Parameter Values : PriceSettingLevel [3]

Pricelist	Price Level
Global	Gross / Net
Germany	Gross / Net
France	Gross

### StrategyDefinition PP

Column name	Strategy Name	Level	Calculation Engine	Additional Engine Configuration	StrategyCalculationParameters	Independent Level Only	Independent Level Priority	Overridable
Values	{user friendly name of the strategy}	<ul style="list-style-type: none"> <li>• Independent</li> <li>• Dependent</li> </ul>	<ul style="list-style-type: none"> <li>• {name of the predefined engine} or</li> <li>• {path to the function in groovy Library which will perform calculation}</li> </ul>	{name of PP with the engines customization}	{list of parameter names which are sent to calculation}	<ul style="list-style-type: none"> <li>• Yes</li> <li>• No</li> </ul>	<ul style="list-style-type: none"> <li>• Yes</li> <li>• No</li> </ul>	<ul style="list-style-type: none"> <li>• Yes</li> <li>• No</li> </ul>

			<p>tions}</p> <p>To learn about the built-in pricing engines, see <a href="#">Price Strategies</a>.</p>					
<b>Description</b>	e.g. "CostPlus"	Determines if the strategy can be calculated at independent /dependent level. If you need it for both, then create two entries with different values here.	Path should look like this: "libs.MyLib.MyElement.MyFunction"	Not all engines are customizable, so this field is nullable. Moreover, passing configuration only to the predefined engines is supported.	<p>Default parameters are:</p> <ul style="list-style-type: none"> <li>• SKU</li> <li>• TARGET_DATE</li> <li>• PRODUCT_COST</li> <li>• BASE_PRICE</li> <li>• LOOKUP_KEYS</li> <li>• PLUS_FOR_PRODUCT</li> <li>• COMPETITION_PRICES</li> <li>• MINIMUM_MARGIN_PRICE</li> <li>• PRICE_INCREASE</li> <li>• DEPENDENCY_INFORMATION_VALUES</li> <li>• BOM_LIST</li> <li>• DISCOUNTS</li> </ul> <p>If additional parameters are necessary, they need to be added from the code to the CalculatedPrices element in the "additionalParameters" map.</p> <p> The parameters are passed as an input to engines, so the order is important and should not be changed for default engines.</p>	If Yes, independent level results of this strategy will not be shown on dependent level PG/PL.	If Yes, this strategy will be above the dependent level results on a dependent level PG/PL.	If No and this strategy is the most important for the product represented by the given lookup keys, it is not possible to override the price through exception tables and manual overrides, regardless of exception configuration.

Example:

Price Parameter Values : StrategyDefinition [18]							
Strategy Name	Level	Calculation Engine	Additional Engine Configuration	Strategy Calculation Parameters	Independent Level Only	Independent Level Priority	
<input type="checkbox"/> RRP	Independent	LookupEngine	RRPLookupEngineConfig	SKU,TARGET_DATE,COUNTRY_INFORMATION_VALUES			
<input type="checkbox"/> RRP	Dependent	LookupEngine	RRPLookupEngineConfig	SKU,TARGET_DATE,COUNTRY_INFORMATION_VALUES			
<input type="checkbox"/> PriceIncrease	Independent	AdjustmentEngine	PriceIncreaseAdditionalConfig	BASE_PRICE,PRICE_INCREASE			
<input type="checkbox"/> PriceIncrease	Dependent	AdjustmentEngine	PriceIncreaseAdditionalConfig	BASE_PRICE,PRICE_INCREASE			
<input type="checkbox"/> MinCompetition	Independent	CompetitionEngine	MinCompetitionAdditionalConfig	COMPETITION_PRICES			
<input type="checkbox"/> MinCompetition	Dependent	CompetitionEngine	MinCompetitionAdditionalConfig	COMPETITION_PRICES			
<input type="checkbox"/> MaxCompetition	Independent	CompetitionEngine	MaxCompetitionAdditionalConfig	COMPETITION_PRICES			

## VolumeBreakdownExceptions PP

For additional details see also [Volume Breakdown](#).

Column name	SKU	Dependent Level Name	Volume #01...#15	Discount #01...#15
Values	{sku from Product master}	{name of dependency level} as stated in <a href="#">DependencyConfiguration PP</a>	Value representing the product volume that will use the given discount	Discount for the given volume
Description	e.g. "MB-0001"	e.g. "Global"	<p>Values here represent at what volumes the discount should start to be applied, e.g.</p> <ul style="list-style-type: none"> <li>• #01 = 5</li> <li>• #02 = 10</li> <li>• #03 = 15</li> </ul> <p>Technically, we use these values just for showing them on a price list and applying a discount, so if prices from this package are used by some quoting solution, it has to apply a proper filtering based on the quoted volume.</p> <p>If one of these values is missing, its volume-discount pair will be ignored.</p>	<p>E.g. 15% These discounts apply to ranges of volumes defined in the Volume columns.</p> <p>If one of these values is missing, its volume-discount pair will be ignored.</p>

Example:

Price Parameter Values : VolumeBreakdownExceptions [1]								
SKU	Dependency Level Name	Volume 1	Discount 1	Volume 2	Discount 2	Volume 3	Discount 3	
<input type="checkbox"/> MB-0007	Global	5	10.00 %	15	20.00 %	25	30.00 %	

## WarningConfig PP

For additional details, see also [Warning Handling](#).

Column name	Key1	Key2	Message	Solution	Type	Alert	Matrix	Warning

<b>Values</b>	{error code}	<ul style="list-style-type: none"> <li>• *</li> <li>• {DependencyLevelName}</li> </ul>	Any text	Any text	Any text	<ul style="list-style-type: none"> <li>• Empty</li> <li>• "Message"</li> <li>• "Yellow"</li> <li>• "Red"</li> <li>• "Critical"</li> </ul>	<ul style="list-style-type: none"> <li>• Yes</li> <li>• No</li> </ul>	<ul style="list-style-type: none"> <li>• Yes</li> <li>• No</li> </ul>
<b>Description</b>	Code of the warning, e.g. "NO_DIMENSIONS".  See their <a href="#">full list</a> .	Defines which dependency this configuration is for. There should be a fallback for each error code by leaving an entry with an asterisk. This field may be ignored if every dependency shares same warning handling.	Message displayed to the user describing what went wrong, e.g. "Price parameter called 'PriceSettingDimensions' does not exist or is empty."	Solution displayed to the user describing how to fix the issue. It is shown only in a ResultMatrix.	Type of the issue displayed to the user. It is shown only in a ResultMatrix, e.g. "Business issue"	Defines how the warning will be classified: on the element level or with an increasing severity.  We advise against using the critical alert too often, as it overrides the color for the whole item. It does not quite fit with the corridor configuration from the price checks module.	Defines if the warning will be registered in a ResultMatrix.	Defines if the warning will be registered in the default Warnings column.


Example:

**Price Parameter Values : WarningConfig [58]**

<input type="checkbox"/>	Key1	Key2	Message	Solution	Type	Alert	Matrix
<input type="checkbox"/>	CANT_GET_REASON	*	CANT_GET_REASON		Other	Red	Yes
<input type="checkbox"/>	CANT_READ_DATA_FOR...	*	CANT_READ_DATA_FOR...		Error	Red	Yes
<input type="checkbox"/>	CANT_READ_DISCOUNT	*	CANT_READ_DISCOUNT		Runtime	Red	Yes
<input type="checkbox"/>	DEPENDENCY_ADJUST...	*	DEPENDENCY_ADJUST...		Data		Yes
<input type="checkbox"/>	ERROR_GETTING_INDE...	*	ERROR_GETTING_INDE...		Error		Yes
<input type="checkbox"/>	ERROR_LOOKING_UP_S...	*	ERROR_LOOKING_UP_S...		Error	Red	Yes
<input type="checkbox"/>	ERROR_LOOKING_UP_C...	*	ERROR_LOOKING_UP_C...		Error	Red	Yes
<input type="checkbox"/>	ERROR_LOOKING_UP_D...	*	ERROR_LOOKING_UP_D...		Error	Red	Yes
<input type="checkbox"/>	ERROR_LOOKING_UP_...	*	ERROR_LOOKING_UP_...		Error	Red	Yes
<input type="checkbox"/>	ERROR_LOOKING_UP_V...	*	ERROR_LOOKING_UP_V...		Error	Red	Yes
<input type="checkbox"/>	ERROR_LOOKING_UP_R...	*	ERROR_LOOKING_UP_R...		Error	Red	Yes
<input type="checkbox"/>	ERROR_PARSING_VOLU...	*	ERROR_PARSING_VOLU...		Error	Red	Yes
<input type="checkbox"/>	EXCEPTION_IGNORED	*	EXCEPTION_IGNORED		Other	Red	Yes
<input type="checkbox"/>	EXCEPTION_STRATEGY...	*	Exception overwritten	Do you really want to overr...	Business Warning	Yellow	Yes
<input type="checkbox"/>	NO_CORRIDOR_CONFIG	*	NO_CORRIDOR_CONFIG		Error		Yes
<input type="checkbox"/>	NO_COST_ENTRY_IN_C...	*	NO_COST_ENTRY_IN_C...	Check Data	Data		Yes
<input type="checkbox"/>	NO_DEPENDENCY_LEV...	*	Missing Dependency Lev...	Define Dependency Level ...	Configuration	Critical	Yes
<input type="checkbox"/>	NO_DEPENDENCY_ADJ...	*	NO_DEPENDENCY_ADJ...		Data	Red	Yes
<input type="checkbox"/>	NO_DISCOUNT_VALUE	*	NO_DISCOUNT_VALUE	Check Discount Data	Data	Red	Yes
<input type="checkbox"/>	NO_EXCHANGE_RATE	*	NO_EXCHANGE_RATE		Data	Red	Yes
<input type="checkbox"/>	NO_FINAL_PRICE	*	NO_FINAL_PRICE		Error	Critical	Yes

**CompetitionAdditionalConfig PP**

Column name	Name	Mode
Values	{Setting Name}	{Setting Value}
Description	To learn about the built-in pricing engines, see <a href="#">Calculation Engines</a> .	

 This is just a sample strategy. You can make as many copies of this PP as you want. You insert the used PP name in a PP called "StrategyDefinition". You can even skip this step if you do not use Competition Based Pricing. Configuring other calculation engines will work in a similar fashion.

Example:

<b>Price Parameter Values : CompetitionAdditionalConfig [5]</b>		
<input type="checkbox"/>	Name	Mode
<input type="checkbox"/>	Competitor Position	min
<input type="checkbox"/>	Repositioning %	
<input type="checkbox"/>	Force Margin Check	no
<input type="checkbox"/>	Repositioning Abs	+10
<input type="checkbox"/>	Price Position	

## Error Handling Deep Dive

Error handling is structured. All errors should be expected and they are all defined in <https://pricefx.atlassian.net/wiki/pages/resumedraft.action?draftId=2592702511> with instructions what to do when such an error happens.

### Limitations

Not everything can be covered by warnings due to technical reasons - the warning manager itself needs some data to be initialized. There are two groups of deployment issues which are not configurable - they raise an exception which is not caught:

1. Lack of code which is required for Price Setting Package to run:
  - a. PriceBuilderCommonElementUtils
  - b. PriceListManagement
  - c. SharedLib
2. Lack of the most basic price parameters:
  - a. PriceSettingConfig
  - b. DependencyConfiguration
  - c. DependencyMappingConfig
  - d. WarningConfig

### Unexpected Errors

These are also two steps for "default" exceptions in case something unexpected happens:

1. "UNEXPECTED\_ERROR" entry in the [WarningConfig PP](#). This is manageable by users but we strongly recommend to set it to "Critical, Yes, **Yes**" and report every occurrence of such an error. When this error is used, it has a modified message to help track what went wrong.
2. Undefined behavior is specified in the code and it has the following code: "NO\_ERROR\_DEFINED".

### Abortable Errors

There is a group of errors that tells us that there is no point in doing any further calculations. We would not be able to get any valid results either way, so we abort the calculation instead of passing on wrong /missing data/configurations. The calculation is aborted on a module-level, so only elements in the module that raised one of these errors are skipped.

The list of abortable error types:

- VALIDATION
- MODULE\_UNUSABLE
- NO\_CONFIG
- EMPTY\_CONFIG

### Caching Lookup Results

The calculation logic contains multiple lookups from both raw user data (like Datamarts, Data Sources or Product Extensions tables) and calculation specific configuration in Price Parameters. Some of it is cached

by default (e.g. sales and forecast data or configuration from Price Parameters), but some of is not. Especially configuration tables that are split based on "dimensions" which are described in [Product Segmentation](#).

It depends on the granularity of the product segmentation and the number of products whether caching such a configuration helps the logic execution times.

To address this, you can use this option in Calculation Inputs:


The screenshot shows the 'Calculation Inputs' configuration interface. At the top, there are two unchecked checkboxes: 'Allow distributed calculation' and 'Allow column type change'. Below these are 'Dynamic item mode' (a dropdown menu) and 'Dynamic item filter' (a link to 'Create Filter'). A note states: 'This calculation logic will be used if no specific method is defined in the product master data.' Below the note are several fields: 'Default pricing logic' (dropdown with 'IndependentPric...' and a refresh icon), 'Dynamic UOM' (dropdown), 'Dynamic currency' (dropdown), 'Result Price' (dropdown with 'FinalPrice'), 'Auto-approve' (dropdown), 'Manual Price Expiry' (dropdown), 'Increase Threshold [%]' (text input), and 'Decrease Threshold [%]' (text input). At the bottom, the 'Cache lookup results' checkbox is highlighted with a red rectangular box.

By default it is switched off since it may impact the performance negatively (particularly in cases where the segmentation is big). This should be the first thing to check when looking for optimization.

## Batching

All lookups are batched as described in [Data Lookups](#).

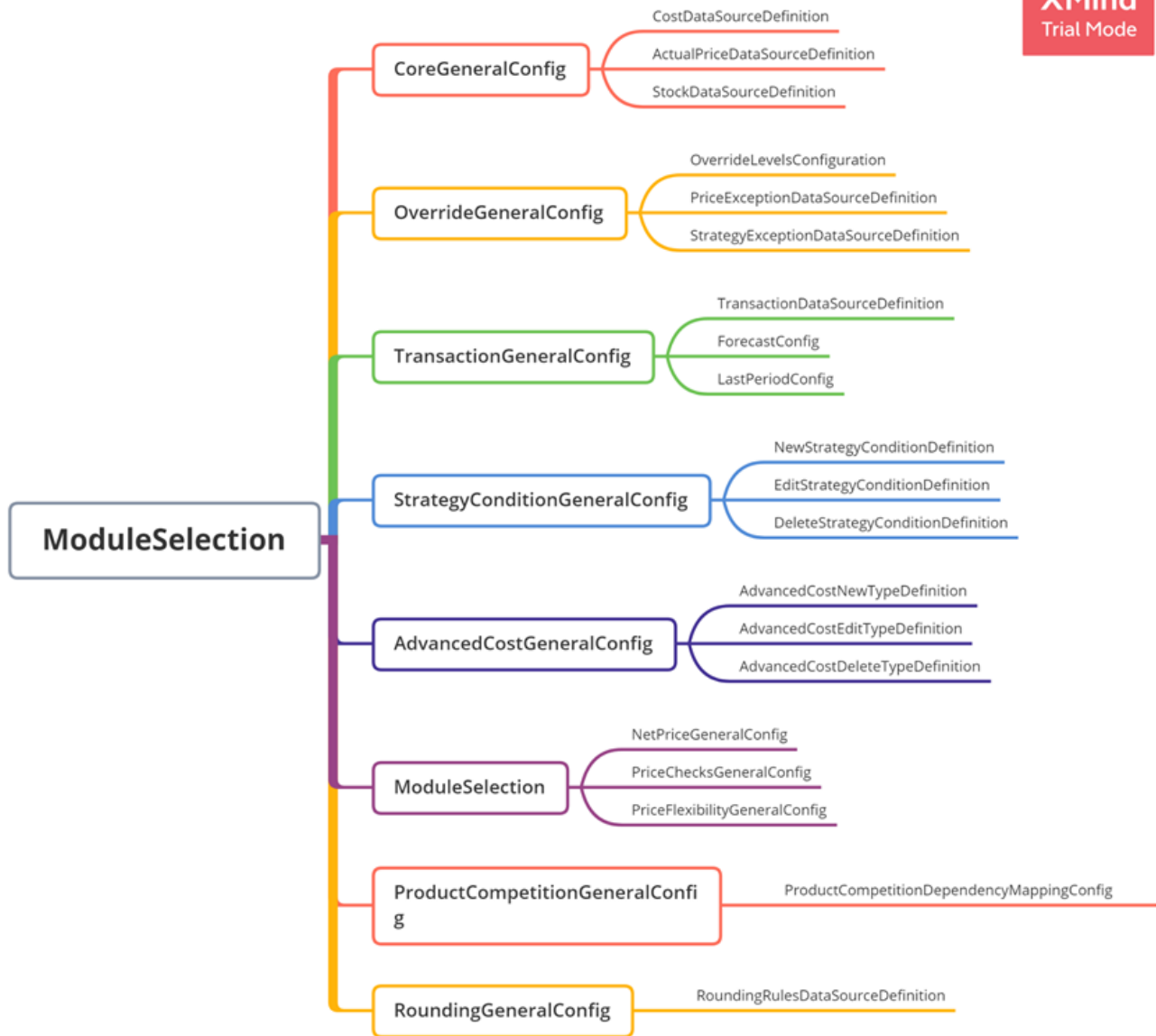
## Price Setting Configuration Wizard - Technical Design

 This document is for maintenance and upgradability. Do not publish it to users.

The business description of the Price Setting Configuration Wizard can be found at [Price Setting Package Configuration Wizard](#).

In this section:

- [Screen Map](#)
- [Components](#)
- [General Calculation Flow](#)
- [PSP\\_ConfigWizardScreenFactory Library Structure](#)
- [PSP\\_ConfigWizardCommonLib Library Structure](#)



**Components**

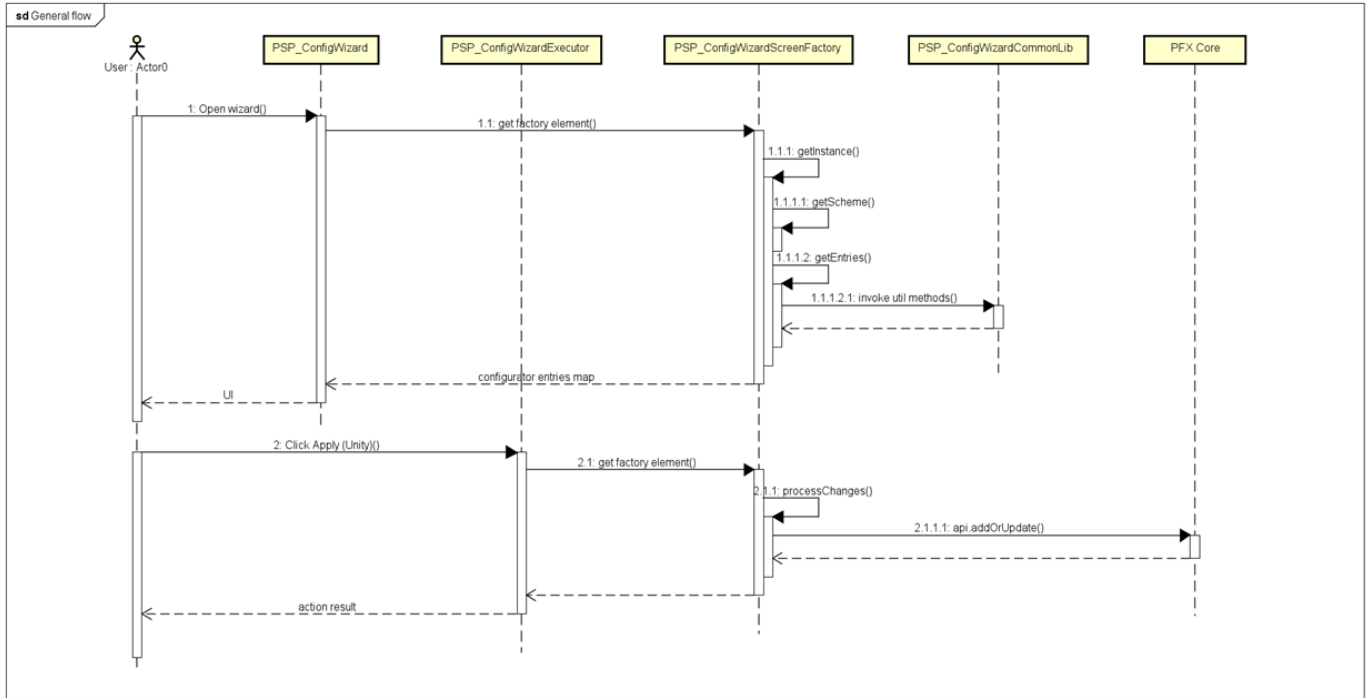
**Generic logics**

- PSP\_ConfigWizard
- PSP\_ConfigWizardExecutor

**Libraries**

- PSP\_ConfigWizardScreenFactory
- PSP\_ConfigWizardCommonLib

## General Calculation Flow



## PSP\_ConfigWizardScreenFactory Library Structure

Each element in the library represents a screen, with proper naming.

#	Name	Label
1	ModuleSelection	
2	CoreGeneralConfig	
3	CostDataSourceDefinition	
4	ActualPriceDataSourceDefini	
5	StockDataSourceDefinition	
6	AdvancedCostGeneralConfig	
7	AdvancedCostGeneralTypeD	
8	AdvancedCostNewTypeDefir	
9	AdvancedCostEditTypeDefini	
10	AdvancedCostDeleteTypeDe	
11	OverrideGeneralConfig	
12	PriceExceptionDataSourceDe	
13	OverrideLevelsConfiguration	
14	ExceptionDataSourceDefiniti	

Syntax to get a factory element: `Script screen = libs.PSP_ConfigWizardScreenFactory[currentScreen]`

Syntax to get a screen instance: `libs.PSP_ConfigWizardScreenFactory[currentScreen].getInstance()`

Syntax to get a screen DB update handler: `libs.PSP_ConfigWizardScreenFactory[currentScreen].processChanges()`

## Element Structure

Must have these public methods:

- Map `getInstance()` - To produce configurator entries set for the screen from a predefined scheme and handled businesses.
- `void processChanges()` - To update values to DB (if any).

## Scheme Format

Must have `_currentInstanceName` hidden configurator entry, with fixed value as the current screen name.

Must have current screen instance-id hidden configurator entry. This is to determine whether the screen is a first-time run.

Each input has a properties map. Supported properties:

- **type** - Typically `InputType.X`, but can also be null. Null means that the result after rendering is a text line, not an input.
- **label** - String, the input label.
- **defaultValue** - Object, the value of the input on the first time run.
- **value** - Object, fixed value.
- **readOnly** - Boolean, disables the input from entering a value.
- **required** - Boolean, the user has to set the input to a valid value to proceed with the wizard.
- **valueOptions** - List, the options of a dropdown list input or a radio button input.
- **valueLabels** - Map, the labels of the options of a dropdown list input or a radio button input.
- **value labels structure** - [option value 1: option label 1, option value 2: option label 2, ...]
- **noRefresh** - Boolean, to prevent logic from rerun when an input value has changed.
- **message** - String, to show message / HTML string instead of a regular input.

Scheme:

```
Map getScheme() {
    String idInputName = libs.PSP_ConfigWizardCommonLib.SchemeUtilities.getInstanceIDInputName(SCREEN_NAME)
    Script descriptions = libs.PSP_ConfigWizardCommonLib.HTMLDescriptions

    return [_currentInstanceName: [type: InputType.HIDDEN, value: SCREEN_NAME],
            (idInputName)       : [type: InputType.HIDDEN],
            description         : [message: descriptions.MODULE_SELECTION],
            currentConfig       : [message: getCurrentSettingsParagraph()],
            moduleSelector      : [type: InputType.OPTION, label: descriptions.SELECT_MODULE_INPUT, noRefresh: true],
            configButton        : [type: InputType.BUTTON, label: "Configure selected module"]]
}
```

Example

The ModuleSelection element structure is as follows:

```

@Field String SCREEN_NAME = "ModuleSelection"
@Field String CORE_MODULE_NAME = "coreModule"
@Field String CORE_MODULE_LABEL = "Core Elements"

// produce the configurator entries set
Map getInstance() {...}

void processChanges() { // the function is for the executor logic interface implementation }

// for the modules table html
protected Map getModuleMapping() {...}

// for the modules table html
protected String getCurrentSettingsParagraph() {...}

// for the modules table html
protected String getStatusRowsDefinition() {...}

// for the modules table html
protected String getStatusLabel(def moduleStatus) {...}

// the predefined screen scheme
protected Map getScheme() {...}

// implementation of producing configurator entries set
protected Closure getEntries() {...}

// handle navigation button / get target screen entries
protected Closure handleButtonEvents() {...}

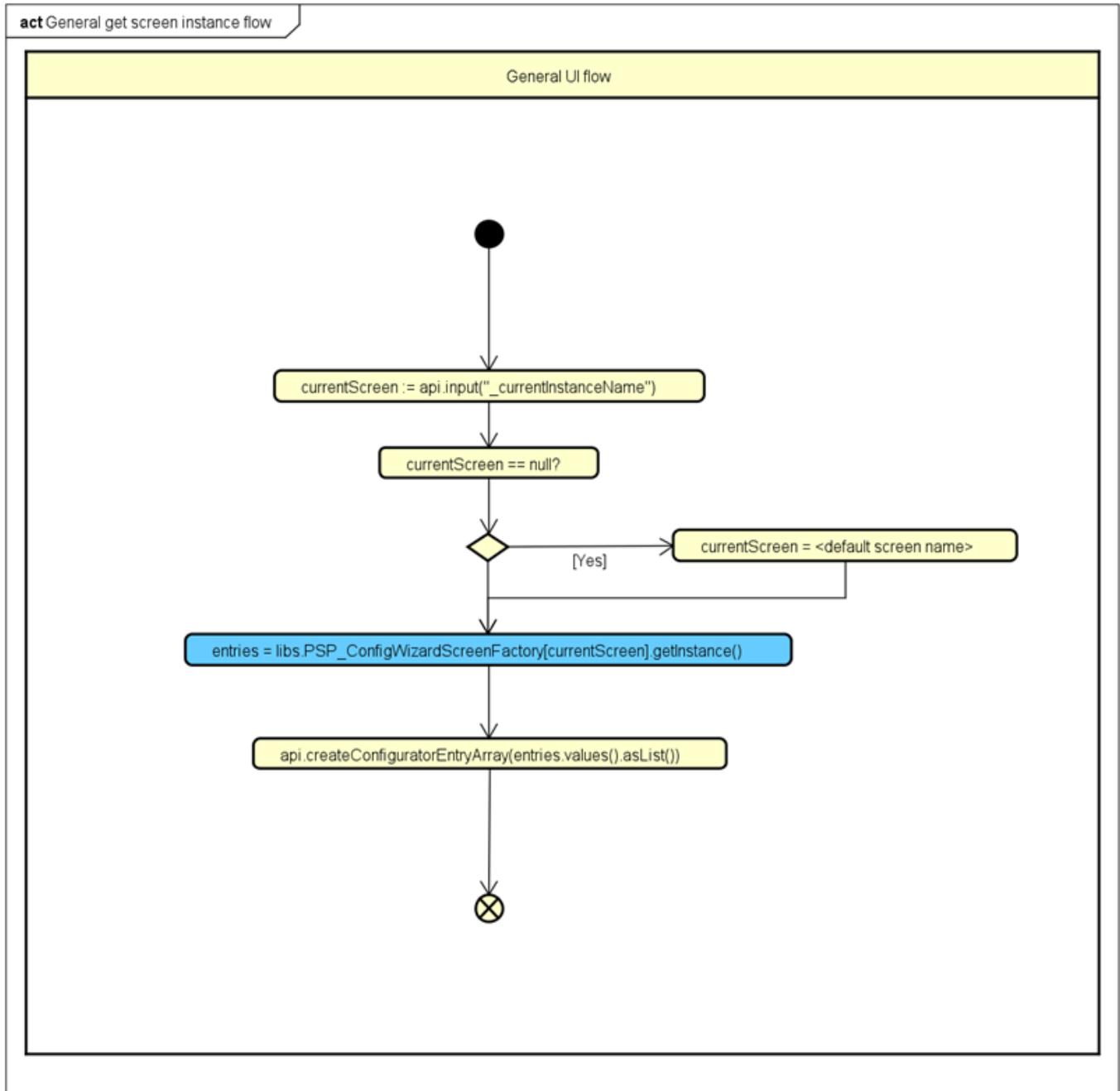
```

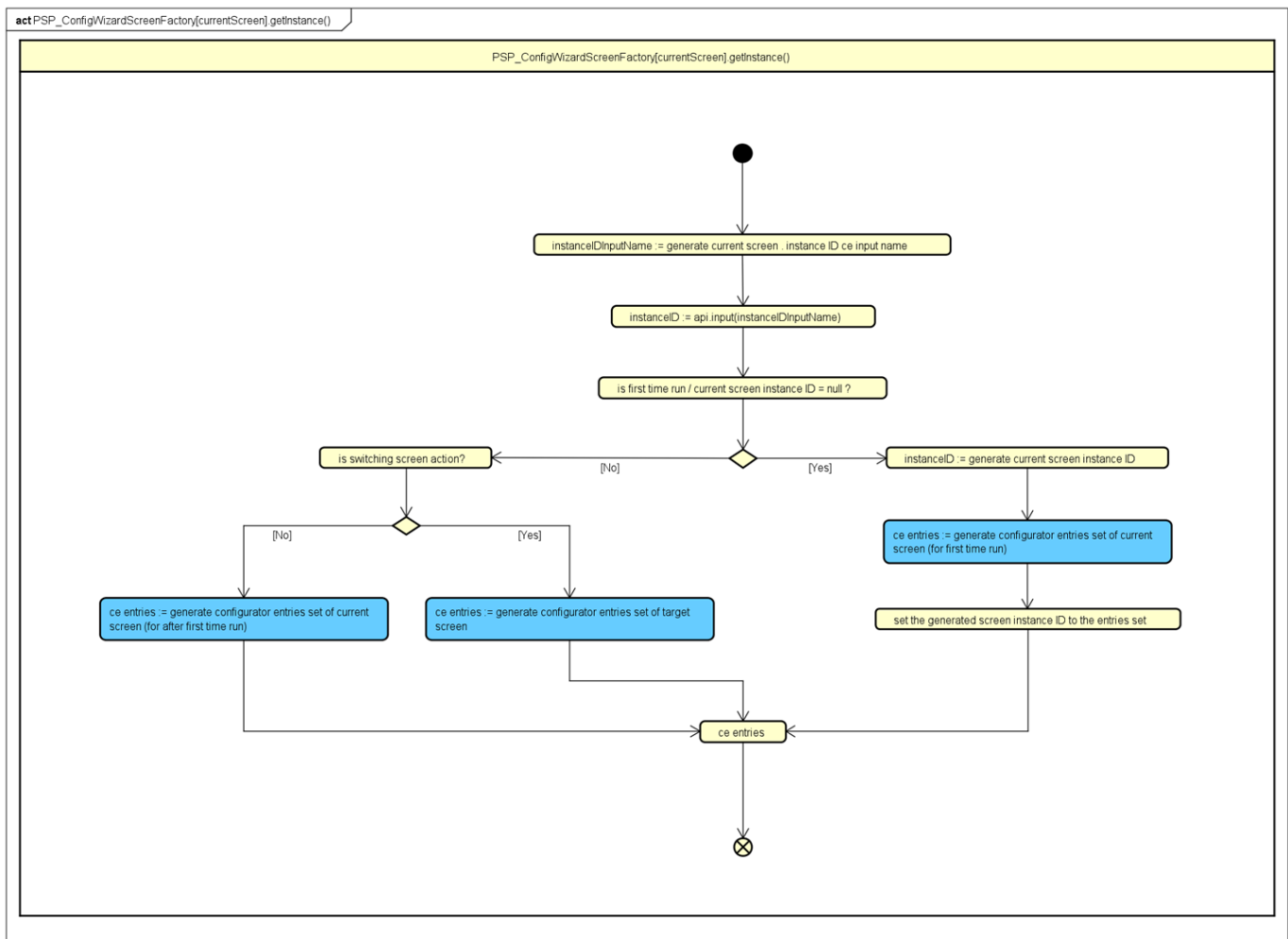
## PSP\_ConfigWizardCommonLib Library Structure

A set of predefined utilities which consists of:

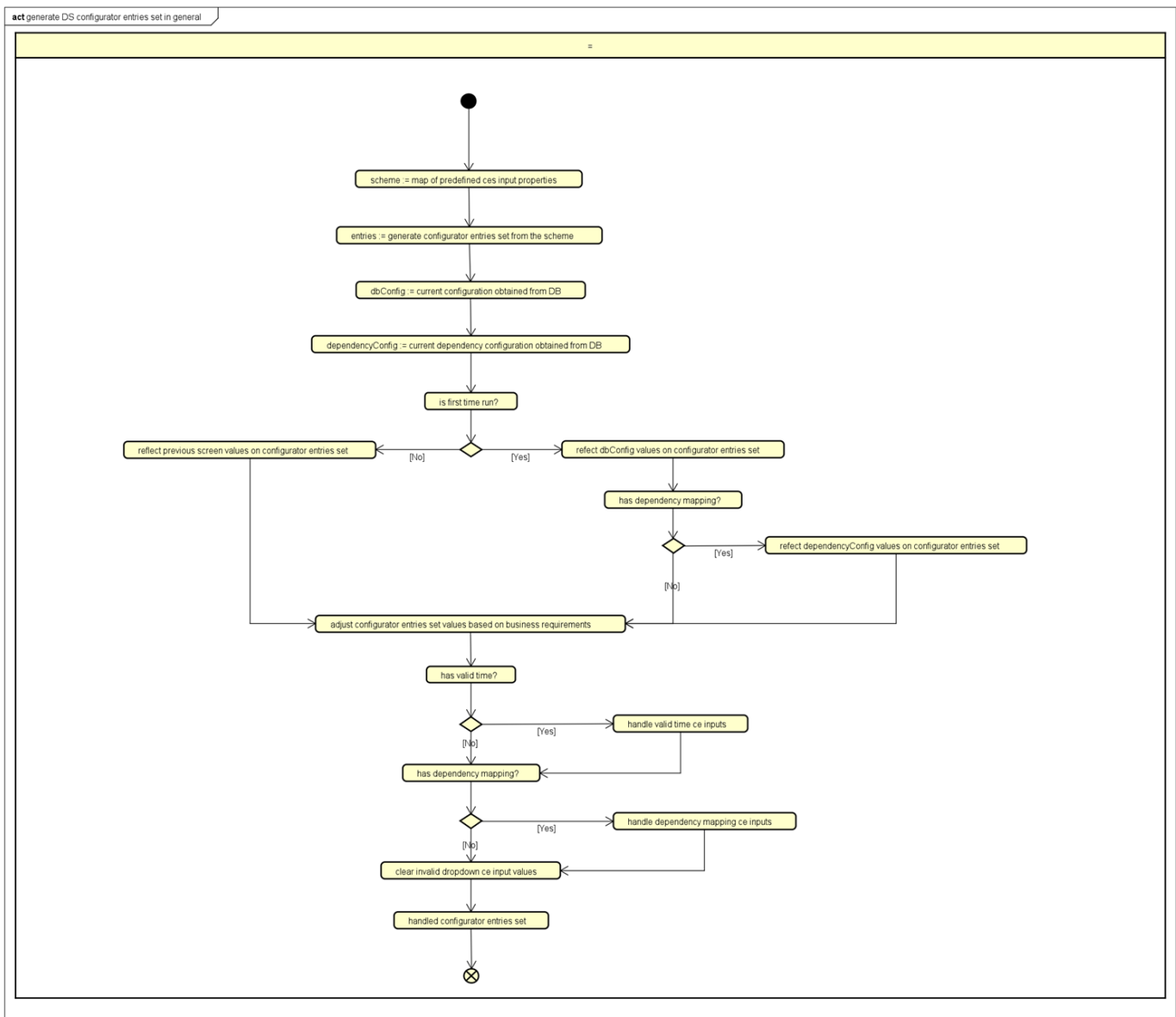
- **Constants** - Contains constants used throughout the wizard such as common input labels,...
- **HTMLDescriptions** - Contains HTML constants used throughout the wizard.
- **HTMLDescriptionUtilities** - Contains HTML handler methods.
- **Errors** - Contains error handler methods (temporarily unused).
- **ScreenStateUtilities** - Contains configurator entry state modification methods, such as set a CE input value, properties,...
- **InputBuilder** - Contains configurator entry builder methods, to build a CE from a predefined scheme.
- **DataUtilities** - Contains database-related methods, such as fetching configuration data rows,...
- **SchemeUtilities** - Contains methods for common scheme manipulation/parsing, with business handling.
- **ModuleUtilities** - Contains methods for common module-related operations, with business handling.
- **NavigationUtilities** - Contains methods for common stuff related to handling screen navigation.
- **InputUtilities** - Contains common methods that provide data for a dropdown input, validate an input value, handle common business operations,.... Typically they are for inputs that have dynamic data /properties based on the user interaction (eg. checkbox selected by user unlocks some input fields).

## Screen Instance Flow





The following chart is for a typical data source configuration screen. Depending on businesses, it may get different between screens.



## Price Setting Webinars and Other Materials

Name	Description	Source
Accelerators in Practice Webinar: Price Setting Package	A business/configuration deep dive into deployment and configuration of Price Setting Package.	<a href="#">Recording</a>

## Price Setting Versions

- [Price Setting Package Release Notes](#)
- [Archive of Previous Versions of Price Setting Package](#)

## Price Setting Package Release Notes

- [Price Setting Package 2.1.0](#)
- [Price Setting Package 2.0.2](#)
- [Price Setting Package 2.0.1](#)
- [Price Setting Package 2.0.0](#)
- [Price Setting Package 1.7.1](#)
- [Price Setting Package 1.7.0](#)
- [Price Setting Package 1.6.0](#)
- [Price Setting Package 1.5.0](#)
- [Price Setting Package 1.4.3](#)
- [Price Setting Package 1.4.2](#)
- [Price Setting Package 1.4.1](#)
- [Price Setting Package 1.4.0](#)
- [Price Setting Package 1.3.0](#)
- [Price Setting Package 1.2.1](#)
- [Price Setting Package 1.2.0](#)
- [Price Setting Package 1.1.1](#)
- [Price Setting Package 1.1.0](#)

## Price Setting Package 2.1.0

- [Manual Changes](#)
  - [Remove Unnecessary Library](#)
  - [Add New Row to WarningConfig Price Parameter](#)
- [Improvements and Fixed Issues](#)
  - [Stories](#)
  - [Improvements](#)
  - [Bugs](#)

### Manual Changes

#### Remove Unnecessary Library

Remove Groovy Library "PriceListManagement". It was extracted to "CalculationEnginesLib" which is now available in PlatformManager, independently from the Price Setting Package.

#### Add New Row to WarningConfig Price Parameter

Error Code	CANT_GET_MARGIN_FLAG
Condition	*
Message	Unable to get the Margin Flag.
Solution	Verify the configuration table and its data.
Type	Data
Alert	Red

Matrix	Yes
Warning	Yes

## Improvements and Fixed Issues

### Stories

[PFPCS-5070](#) Margin Flag

[PFPCS-5069](#) Changing (corridor) colors

[PFPCS-5068](#) Competitive data retrieval: please add date filter

[PFPCS-4110](#) Make calculation engines independent from PSP version

[PFPCS-1602](#) Make Dependency Mapping optional

### Improvements

[PFPCS-4269](#) Do not override PP's folders during upgrade package

[PFPCS-3907](#) Add minimal partition version to PSP and adjust documentation URL

### Bugs

[PFPCS-5245](#) Transaction Data - Cannot invoke method contains() on null object

[PFPCS-5043](#) Inconsistent rounding for Sales Data Volumes fields

[PFPCS-4792](#) Strategy Condition - Check Exception does not work properly

[PFPCS-4788](#) Fix Warning messages

[PFPCS-4595](#) Transaction configuration wizard - config in fields where aggregated values are stored can't be cleared properly

[PFPCS-4443](#) Transaction module configuration wizard - Issue with required fields when using PX as source type

## Price Setting Package 2.0.2

### Bugs

[PFPCS-4256](#) Add wizards into upgrade package

[PFPCS-4252](#) Add temporary hooks into upgrade package

[PFPCS-4221](#) PM - Mandatory hierarchy should be required

### Upgrade note:

- Module configuration migrated from AP to PP
  - Name of PP: "PriceSettingModules"
  - Matrix type with 1 key
  - Keys are:
    - PSP\_ADVANCED\_COST
    - PSP\_NET\_PRICE\_MODULE
    - PSP\_OVERRIDES\_MODULE
    - PSP\_PRICE\_CHECKS\_MODULE

- PSP\_PRICE\_FLEXIBILITY\_MODULE
- PSP\_PRODUCT\_COMPETITION\_MODULE
- PSP\_ROUNDING\_RULES\_MODULE
- PSP\_STRATEGY\_CONDITION\_MODULE
- PSP\_TRANSACTION\_MODULE
- Values are:
  - On
  - Off

## Tasks

[PFPCS-4218](#) Fix PF -> PSP integration

## Price Setting Package 2.0.1

### Fixed Bugs

[PFPCS-4143](#) Attribute-Based Engine - error about dependency mapping is thrown when source type is P /PREF

[PFPCS-4134](#) STRATEGY\_NAME parameter is the same across all strategies

## Price Setting Package 2.0.0

- [Manual Changes](#)
  - [Before Running Upgrade PlatformManager Script](#)
  - [What If Price Setting Package Was Already Upgraded?](#)
- [Improvements and Fixed Issues](#)
  - [Stories](#)
  - [Improvements](#)
  - [Tasks](#)
  - [Bugs](#)

## Manual Changes

### Before Running Upgrade PlatformManager Script

- PriceSettingDimensions:
  - Key3 must be added. Put the string "Fallback" in all entries (it must be done, no matter if you wish to use this feature or not).
  - If you want to override lookup keys of some of the lookups, add new entries to PriceSettingDimensions.
    - Key1 - "Products"
    - Key2 - {proper number, from 1 to 5}
    - Key3 - String, based on which lookup config you want to override. Possible values (some of them is behind module) are:
      - Fallback (used when per-feature key is not defined)
      - StrategySelection
      - BaseStrategySelection
      - MinMargin
      - CostPlus
      - PriceIncrease
      - AdditionalDiscount
      - RelevantCompetitors
      - DependencyAdjustment

- VolumeBreakdown
- AdjustedPriceCorridor
- ListPriceCorridor
- CostSelection
- Discount
- Dependency Configuration:
  - New column named "isComplete. It requires that the user moves all preferences manually:
    - All columns names, as preferences, must be manually moved 1 column to the right (there is 1 preference less in the end).
    - For more detailed description, see the upgrade note of PFPCS-1915.

#### What If Price Setting Package Was Already Upgraded?

What to do if you already ran the upgrade of Price Setting Package before applying the manual changes above? Then Accelerator's upgrade should fail. Do these changes now and run the package again.

#### Improvements and Fixed Issues

##### Stories

[PFPCS-3512](#) Update eCommerce standard config

[PFPCS-3309](#) Move module configuration to Price Parameter

[PFPCS-3270](#) Return message in Competition Pricing - Price Position Mode

[PFPCS-3241](#) Add Configuration Wizard

[PFPCS-2996](#) Add Module information to warnings popup

[PFPCS-2995](#) Show severity and color coding in warning popup

[PFPCS-2994](#) Add default messages and solutions to WarningConfig

- **Upgrade note:**
  - Changes to WarningConfig Price Parameter:
    - Multiple messages updated (QoL, but it will not be applied to users who upgrade the package). Default values have been listed [here](#).
    - Renamed error codes:
      - NO\_COST\_ENTRY\_IN\_COST\_TABLE NO\_COST\_FOUND
      - ERROR\_LOOKING\_UP\_DEPENDENCY\_LEVEL\_ADJUSTMENT CANT\_GET\_DEPENDENCY\_LEVEL\_ADJUSTMENT
      - NO\_FINAL\_PRICE INVALID\_FINAL\_PRICE
      - INVALID\_DIMENSIONS INVALID\_DIMENSION\_LOOKUP\_FIELD
      - NO\_EXCHANGE\_RATE NO\_EXCHANGE\_RATE\_FOUND
      - ERROR\_LOOKING\_UP\_COST\_SELECTION\_CONFIG CANT\_GET\_COST\_TYPE\_SELECTION
      - ERROR\_PARSING\_COST\_DEFINITION ERROR\_PARSING\_COST\_TYPE\_DEFINITION
      - ERROR\_CALCULATING\_COST CANT\_CALCULATE\_COST\_TYPE
      - CANT\_READ\_DISCOUNT CANT\_GET\_DISCOUNT
      - NO\_DISCOUNT\_VALUE NO\_DISCOUNT\_FOUND
      - NO\_ACTUAL\_LIST\_PRICE NO\_ACTUAL\_LIST\_PRICE\_FOUND
      - NO\_PL\_FOR\_PRICE\_LOOKUP CANT\_GET\_ACTUAL\_PRICE\_FROM\_PL
      - INVALID\_DEPENDENCY\_LEVEL\_TABLE\_ID INVALID\_INDEPENDENT\_SOURCE\_ID
      - NO\_SUPPORTED\_TABLE\_TYPE UNSUPPORTED\_INDEPENDENT\_SOURCE\_TYPE
      - NO\_INDEPENDENT\_LEVEL\_ITEM NO\_INDEPENDENT\_LEVEL\_RECORD\_FOUND
      - NO\_INDEPENDENT\_LEVEL\_DECISION NO\_INDEPENDENT\_LEVEL\_PRICE\_DECISION

- ERROR\_GETTING\_INDEPENDENT\_LEVEL\_PRICES\_FROM\_INDEPENDENT\_LEVEL\_ITEM NO\_INDEPENDENT\_LEVEL\_CALCULATED\_PRICES
- ERROR\_LOOKING\_UP\_CORRIDOR\_CONFIG CANT\_GET\_CORRIDOR\_CONFIG
- NO\_CORRIDOR\_CONFIG NO\_CORRIDOR\_CONFIG\_FOUND
- CANT\_GET\_REASON CANT\_GET\_PRODUCT\_CHANGED\_CAUSE
- NO\_REASON\_FOR\_PF NO\_PRODUCT\_CHANGED\_CAUSE
- ERROR\_LOOKING\_UP\_MINIMUM\_MARGIN CANT\_GET\_MIN\_MARGIN
- NO\_INPUT\_FOR\_MIN\_MARGIN\_HANDLING  
NO\_INPUT\_FOR\_MIN\_MARGIN\_VALIDATION
- ISSUE\_EXECUTING\_STRATEGY\_CONDITION CANT\_APPLY\_STRATEGY\_CONDITION
- ERROR\_LOOKING\_UP\_STRATEGIES CANT\_GET\_STRATEGIES\_SELECTION
- NO\_STRATEGY\_DEFINITION\_ENTRY NO\_STRATEGY\_DEFINITION\_FOUND
- ERROR\_LOOKING\_UP\_RELEVANT\_COMPETITION\_DATA\_PP  
CANT\_GET\_RELEVANT\_COMPETITION\_DATA
- NO\_ROUNDING\_RULE NO\_ROUNDING\_RULE\_FOUND
- INVALID\_STOCK\_CONFIGURATION CANT\_GET\_STOCK\_CONFIG
- NO\_SOURCE\_TABLE NO\_TRANSACTION\_SOURCE\_TABLE
- NO\_FORECAST\_TYPE INVALID\_FORECAST\_TYPE
- ERROR\_PARSING\_VOLUME\_DISCOUNTS  
INVALID\_VOLUME\_BREAKDOWN\_SETTING
- NO\_INPUT\_FOR\_VOLUME\_DISCOUNT CANT\_GET\_VOLUME\_DISCOUNT
- Business changes:
  - NO\_MIN\_MARGIN\_PRICE (split into) NO\_MIN\_MARGIN\_PRICE and NO\_INPUT\_FOR\_MIN\_MARGIN\_PRICE
  - NO\_MIN\_MARGIN and ARITHMETIC\_EXCEPTION (merged and split based on business case) INVALID\_MIN\_MARGIN\_PERCENT and NO\_MIN\_MARGIN\_CONFIG\_FOUND
  - EXCEPTION\_IGNORED (split into) EXCEPTION\_IGNORED and STRATEGY\_OVERRIDE\_PROHIBITED
  - UNABLE\_TO\_CALCULATE\_BREAKEVEN\_FOR\_DATA (split into) CANT\_CALCULATE\_MARGIN\_BREAKEVEN\_VOLUME and CANT\_CALCULATE\_REVENUE\_BREAKEVEN\_VOLUME
  - PX\_VALIDITY\_PERIODS\_OVERLAPPED and PP\_VALIDITY\_PERIODS\_OVERLAPPED (merged into) VALIDITY\_PERIODS\_OVERLAPPED
  - NO\_DATA\_FOR\_LAST\_PERIOD\_LOOKUP (split into) NO\_SALES\_VOLUME\_LAST\_PERIOD and NO\_TURNOVER\_LAST\_PERIOD
  - CANT\_READ\_DATA\_FOR\_PRICE\_LOOKUP (merged into) NO\_ACTUAL\_LIST\_PRICE\_FOUND
  - UNABLE\_TO\_READ\_TABLE\_DATA No name changed, but now is responsible for all lookups, not only Exception Lookup
- Removed:
  - DISCOUNT\_GROUP\_FALLBACK\_APPLIED (Discount Group Lookup removed)
  - ERROR\_LOOKING\_UP\_DISCOUNT\_GROUP (Discount Group Lookup removed)
  - NUMBER\_FORMAT\_EXCEPTION (not used even in 1.7.1, leftover found)
  - NO\_STRATEGY\_OVERRIDE\_ALLOWED (not used even in 1.7.1, leftover found)
  - NO\_SOURCE\_TYPE (not used even in 1.7.1, leftover found)
  - NO\_DEPENDENCY\_LEVEL\_NAME (not used even in 1.7.1, leftover found)
  - ERROR\_LOOKING\_UP\_VOLUME\_EXCEPTION (not used even in 1.7.1, leftover found)
  - UNABLE\_TO\_READ\_COST\_TABLE (no data don't throw anymore, other cases handled in more general CANT\_CALCULATE\_COST\_TYPE /UNABLE\_TO\_READ\_TABLE\_DATA)

- UNABLE\_TO\_READ\_STOCK\_TABLE and UNSUPPORTED\_STOCK\_DATA\_SOURCE\_TYPE (no data thrown anymore, other cases handled in more general NO\_STOCK\_DATA /UNABLE\_TO\_READ\_TABLE\_DATA)
- UNSUPPORTED\_EXCEPTION\_TYPE (no data thrown anymore, other cases handled in more general UNABLE\_TO\_READ\_TABLE\_DATA)
- Added:
  - NO\_BASE\_DEPENDENCY (part of new feature, multi-level hierarchy)
  - CIRCULAR\_DEPENDENCY (part of new feature, multi-level hierarchy)

[PFPCS-2947](#) Remove spaces from bootstrapped parameters

[PFPCS-2525](#) Change AnchorEngine to proxy for AttributeEngine

[PFPCS-2233](#) Unification of configurable lookups

[PFPCS-1915](#) Add support for multiple groups of Independent/Dependent levels

- **Upgrade note:**
  - New way of handling dependency levels. Please refer to the [Dependent Price Lists and Data Fallbacks](#) in PSP documentation.
  - Data from old DependencyConfiguration can be mostly reused, but data structure is new. Please refer to [DependencyConfiguration PP](#).
    - New column in DependencyConfiguration.
    - All preferences must be manually moved 1 column to the right. New column is called "IsComplete". It should be set to "Yes" or "No".
    - If users want to keep experience as close to the old behavior as possible, they should:
      - Keep proper dimensions to force HQ/non-HQ modes.
      - Set IsComplete to "No".

[PFPCS-1675](#) Positioning in competitors

[PFPCS-1221](#) Adjustments for easier upgradeability and documentation

[PFPCS-1129](#) Support for Table-Level dimension configuration

- **Upgrade note:**
  - These price parameters are changing also due to PFPCS-1902. Please read upgrade notes of both tasks before proceeding.
  - Except pre-upgrade instructions, no further work needs to be done, if users do not want to change behavior of installation.
  - If users override some of the lookup keys configs, a new PP has been created.
    - Config needs to be revised. If the number of new keys is the same as the number of old keys, data should persist. It should be removed and applied from scratch.
    - Column name will be changed, but column label might not. It should be applied manually in such scenario.

[PFPCS-972](#) Show "Lookup Key" in Calculation Logic

- **Upgrade note:**
  - Two new columns in both Independent and Dependent logics. Config needs to be updated for these columns to be visible.

[PFPCS-659](#) Adjust features to Manhattan release

[PFPCS-277](#) (Collection) Small Improvements and Issues in Price Setting

Improvements

[PFPCS-3722](#) Make all tables optional when using Table DependencyMapping

[PFPCS-3622](#) Don't use magic String (""") for dependencyHierarchy in PSP

[PFPCS-3252](#) Check if core warnings might be removed

[PFPCS-1902](#) Make more hierarchical tables per-dependency

- **Upgrade note:**

- These price parameter are changing also due to PFPCS-1129. Please read upgrade notes of both tasks before proceeding.
- Discount group lookup has been removed. With this task it is not needed, as hierarchical lookups might already be differentiated per dependency level.
- Except of discount group, no more manual tasks should be needed, since old price parameters remain as fallbacks for each dependency. As long as no new entry will be added to newly created PP, behavior should not change.

[PFPCS-1807](#) Sorting for Competition Data matrix

#### Tasks

[PFPCS-3891](#) Reorganize PSP documentation

[PFPCS-3755](#) Switch LookupManager usage to Sharedlib's implementation

[PFPCS-3753](#) Switch PPUtil usage to SharedLib's ConfigReaderUtil

[PFPCS-2739](#) Allow calling warningMager.handleWarning() inside batch action

[PFPCS-2628](#) Merge price-builder-accelerator-library with price-builder-accelerator

[PFPCS-1598](#) Change element results after aborting calculation

#### Bugs

[PFPCS-3987](#) Make Sales Volume Forecast consistent with other volumes

[PFPCS-3971](#) Update Strategy definition in standard config

[PFPCS-3967](#) Should not show null price in Competition Data column and Relevant Competition Data column

[PFPCS-3966](#) Configuration issues in ecommerce package

[PFPCS-3965](#) Calculation of Turnover Forecast

[PFPCS-3959](#) When editing condition, Condition right operand will be cleared if current condition is multiplying by number that has decimal

[PFPCS-3939](#) Strategies didn't return results

[PFPCS-3839](#) DEPENDENCY\_LEVEL\_CONFIG in ConstConfig element contains invalid PREFERENCE keys

[PFPCS-3834](#) Advanced Cost - Cost Type should occur only once

[PFPCS-3823](#) Revenue Breakeven Volume is not rounded up correctly

[PFPCS-3811](#) Add table dependency level mapping config placeholder for PriceIncrease and CostPlus

[PFPCS-3805](#) LookupEngine strategy returns error if Source Type PP is used

[PFPCS-3804](#) Secondary Key is empty if the Sku is not defined in PP VolumeBreakdown

[PFPCS-3754](#) Breakeven volumes are not rounded

[PFPCS-3651](#) Add UI object steps.json

[PFPCS-3643](#) Clean up assembly-pom.xml

[PFPCS-3623](#) Incorrect Final Price is displayed because of rounding issue

[PFPCS-3423](#) Error looking up Competitor prices when country is defined in Competition Data

[PFPCS-2917](#) Input configurator for IndependentPriceListLogic doesn't hide dependent level name on Unity

[PFPCS-2330](#) Overlapped validity dates should use warning instead of errors

[PFPCS-1738](#) Anchor and Kit pricing engine doesn't work on first PL calculation

## Price Setting Package 1.7.1

### *Bugs*

[PFPCS-3282](#) Incorrect information about Skipped Competitors returns when using Competitor Position mode

[PFPCS-3248](#) Unexpected Errors happen when Last Period Transaction is not configured properly

[PFPCS-3247](#) Unexpected Error happens when Transaction Source is PX and name of Source Table is missing

[PFPCS-3246](#) Exception Manager returns an error when Source Table of Price Exception/Strategy Exception is missing/incorrect

[PFPCS-3219](#) Competition Engine: Repositioning is not working when competitor is skipped

[PFPCS-3218](#) Stock can't display value 0

[PFPCS-3217](#) Unexpected Error happens when Source Field of Actual Price in PP PriceSettingConfig is empty/incorrect

[PFPCS-3182](#) Fix warnings for Vesper release

[PFPCS-3176](#) Not showing Message Type for [Competition] (Independent Type Strategy) in Dependent LPG

[PFPCS-3174](#) CalculatedPrices in Dependent LPG return an error if only recalculating Dependent LPG after removing strategy in PP StrategySelection

[PFPCS-3171](#) Fix Reason that is displaying in Prices matrix when Cost/BasePrice is null

[PFPCS-3162](#) 3 static lines in the Price Insights Dashboard shows wrong data

[PFPCS-3143](#) Fix value format in Margin Breakeven Volume and Revenue Breakeven Volume columns

## Price Setting Package 1.7.0

- [Change Requests](#)
- [Bugs](#)
- [Tasks](#)
- [Improvements](#)
- [Sub-tasks](#)
- [Stories](#)

See also [Price Setting Package 1.7.0 Key Features Description](#).

### Change Requests

- [\[PFPCS-2642\]](#) - Results of Strategy Engines and handling of them
- [\[PFPCS-2949\]](#) - Adjust return of Competition Engine
  - **Upgrade notes:**
    - Strategy definition needs to be updated. Now Competition engine should be supplied with below parameter names:

- COMPETITOR\_PRICES,MINIMUM\_MARGIN\_PRICE
- RELEVANT\_COMPETITOR\_PRICES),MINIMUM\_MARGIN\_PRICE

## Bugs

- [PFPCS-1564] - Missing dropdown in PPStrategySelection
- [PFPCS-2829] - Remove invalid character from fields
- [PFPCS-2837] - Fix Stock format to display 2 decimals
- [PFPCS-2913] - Incorrect type in definedPP
- [PFPCS-2921] - PP RelevantCompetitionData - Incorrect dropdown values in Relevant Competitors field
- [PFPCS-2929] - LPG still can be saved when InputConfigurator is not filled on Unity
- [PFPCS-2963] - Strategy Engine can't return Message and Message Type when there is no Price
- [PFPCS-3004] - CostPlus parameter is handled incorrectly if AdjustmentEngine expects absolute value
  - **Upgrade notes:**
    - Adjustment engine now has 3 parameters: "Base price", "percentage increase" and "absolute increase".
      - PRICE\_INCREASE engine parameter has been split into PRICE\_INCREASE\_PERCENTAGE and PRICE\_INCREASE\_ABSOLUTE
        - Suggested configuration change is to swap "PRICE\_INCREASE" String with "PRICE\_INCREASE\_PERCENTAGE,PRICE\_INCREASE\_ABSOLUTE" String
      - PLUS\_FOR\_PRODUCT engine parameter split into PLUS\_FOR\_PRODUCT\_PERCENTAGE and PLUS\_FOR\_PRODUCT\_ABSOLUTE
        - Suggested changes for strategyDefinition are:
          - Suggested configuration change is to swap "PLUS\_FOR\_PRODUCT\_PERCENTAGE" String with "PLUS\_FOR\_PRODUCT\_PERCENTAGE, PLUS\_FOR\_PRODUCT\_ABSOLUTE" String
      - New parameter available for StrategyDefinition: "NULL\_VALUE". Nullable parameters which always will be null might be replaced with that one (it ensures that no extra data is put in always empty column).
      - PriceIncrease and CostPlus PriceParameter now has 2 columns in data instead of one.
        - attribute1 percentage increase
        - attribute2 absolute increase
        - Only one of them will be utilized for given engine
    - Data in below PriceParameter has been multiplied by 100 by formatting change. It needs to be manually recovered to proper state:
      - MinMargin
      - Discount
      - AdditionalDiscount
      - DependentLevelAdjustment
      - CostPlus (attribute1)
      - PriceIncrease (attribute1)
- [PFPCS-3008] - vLookupHierachic wildcard fallback doesn't work when only one key is used.
- [PFPCS-3032] - Execution of DependentPriceListLogic[Library] timed out
- [PFPCS-3090] - Fix error after restructuring and renaming
- [PFPCS-3093] - Can't get value defined for Min Margin and unexpected Error displays in RawMinimumMargin
- [PFPCS-3095] - Result of Competition strategies are replaced by error after recalculating item inside LPG

- [PFPCS-3099] - AdjustmentEngine still returns price when %/Absolute is missed
- [PFPCS-3101] - AdjustmentEngine - Change the name of Absolute column
- [PFPCS-3103] - Cost has issues when Condition is set up separately for independent and dependent price lists
- [PFPCS-3123] - Cost Configuration Wizard - should have ability to delete configuration
- [PFPCS-3125] - Add dropdown to column Value in CostPlusAdditionalConfig and PriceIncreaseAdditionalConfig
- [PFPCS-3129] - Missing Warning message REQUIRED\_INPUTS\_MISSING
- [PFPCS-3141] - Unexpected error INVALID\_FINAL\_PRICE displays after recalculating sku in Matrix LPG
- [PFPCS-3156] - Warnings about PriceSelector and ManualPrice not found in elements
- [PFPCS-3160] - Incorrect results return when calculating Breakeven volume
- [PFPCS-3161] - Fix format of Competitor Price in Reason - Prices matrix
- [PFPCS-3173] - Can't show [StrategyName] (Independent Type Strategy) in the last position in Dependent LPG
- [PFPCS-3178] - MinCompetition returns null with incorrect message when its price equals 0 and it is greater than Minimum Margin Price

#### Tasks

- [PFPCS-2707] - Update Independent Priority Usage
- [PFPCS-2736] - Isolate shared variables between priceManager instance and priceManagerImp
- [PFPCS-3084] - Turn off Advanced Cost Config module by default in standard config
- [PFPCS-3085] - Setup default Product Competition Dependency Mapping

#### Improvements

- [PFPCS-2730] - Breakeven quantity in Pricelist
  - **Upgrade notes:**
    - New entries in Warning config
      - UNABLE\_TO\_CALCULATE\_BREAKEVEN\_FOR\_DATA
      - MISSING\_INPUTS\_MARGIN\_BREAKEVEN\_VOLUME
      - MISSING\_INPUTS\_REVENUE\_BREAKEVEN\_VOLUME
- [PFPCS-2938] - libs function should not rely on data from calculation context.

#### Sub-tasks

- [PFPCS-1092] - Cost+ price calculation
- [PFPCS-1215] - Adjust result price by wrapping existing engine into new custom engine
- [PFPCS-1227] - Split documentation into proper pages

#### Stories

- [PFPCS-1041] - Translations in Calculation dispatcher when calling Strategy Engines for Build-In Strategies
- [PFPCS-1042] - Review accessibility of our Library Functions
- [PFPCS-1658] - Include dropdown value selectors if possible in bootstrapped PPs
- [PFPCS-2815] - Setup sources for dropdown values in bootstrapped PPs
- [PFPCS-2907] - PoC: Config Wizard for Cost-Lookup

### Price Setting Package 1.7.0 Key Features Description

This section summarizes some of the highlighted features of the 1.7.0 release. Please note that some of them were already available in older versions of the package, so check the release notes to see in what version the features were introduced.

- [More Information Returned by Pricing Strategies](#)
- [Attribute Based Pricing](#)
- [Advanced Cost](#)

### More Information Returned by Pricing Strategies

Strategy Engines can now return more information. Besides the price, they can return a text message to explain the calculation result, giving the business user some more insights.

This is what you can return for your Custom Engines:

```
return ['price' : 10,
'message' : 'Cheapest
comopetitor: Media Markt
(11.99€)', 'messageType' :
'Info']
```

The messageType can be the following:

- Info will be “green”
- Warning will be “Yellow”
- Critical will be “Red”

When the engine throws an exception, it will be shown in a “red cell with white text”.

### Attribute Based Pricing

Attribute Based Pricing Strategy allows you to define your pricing strategy based on product attributes. The attributes can represent different pricing impact values. The price of the reference product and the value of the pricing relevant attributes will allow you to define your product price accordingly.

Definition of Anchor Data and Strategy Definition

In the first step you have to define the “Anchor-Follower” relations.

Strategy Definition should look like this:

Price Parameter Values: StrategyDefinition					Mass edit	Mass delete	Export
<input type="checkbox"/>	Strategy Name	Level	Calculation Engine	Additional Engine Configuration	Strategy Calculation Parameters		
	Anchor	Select Value	Search...	Search...	Search...		
<input type="checkbox"/>	AttributeBased	Independent	AnchorEngine	AnchorAdditionalConfig	SKU,FINAL_LIST_PRICE_ELEMENT_NAME,FINAL_PRICE_ELEMENT_NAME		

The “Additional Engine Configuration” is like in the old Anchor Engine definition where to find the Anchor data. It has to look like the following example:

Price Parameter Values: AnchorAdditionalConfig	
Name	Value
<input type="checkbox"/> Search...	<input type="text" value="Search..."/>
<input type="checkbox"/> Source Type	PP
<input type="checkbox"/> Source Name	AnchorData
<input type="checkbox"/> Anchor Field Name	Anchor
<input type="checkbox"/> Sku Field Name	SKU

Select Pricing Relevant Attributes

You can define what attributes are relevant for your pricing strategy.

Price Parameter Values: Pricing Attributes									
Pricing Attribute	Type	Source Type	Source Name	Source Field	Dependency Field	Dependency Mapping ...	Mapping Source Field	ValidFrom Field Name	ValidTo Field Name
<input type="checkbox"/> Product Lifecycle	Value	P	Product Lifecycle	Product Lifecycle					
<input type="checkbox"/> Perceived Value	Interval	PX	PerceivedValue	PerceivedValue	DependencyLevelNa...	Lookup	Country	ValidFrom	

This is how your PP should look like after this step.

We have following options for type:

- Value Convert value to "impact", like red = 2, green = 4, ...
- Direct Use the value of the attribute as impact. E.g. 2 kg, 30 %, ... (only works for "number" type values).
- Interval Define interval boundaries and group to impacts (only works for "number" type values).

Convert Value of Pricing Relevant Attributes to Price Impact Values

Once you defined pricing relevant attributes, you can define the impact of the values each of the attributes will have.

This is how the interval based parameter should look like:

Price Parameter Values: IntervalAttributesConversion			
Pricing Attribute	Pricing Attribute Value From	Pricing Attribute Value To	Price Impact Value
<input type="checkbox"/> Perceived Value	0	30	-3
<input type="checkbox"/> Perceived Value	30	75	2
<input type="checkbox"/> Perceived Value	75	100	3

Note: We could define the "perceived value" attribute as "Direct Value". In this case we could use the percent value directly in the pricing formula afterwards.

Create Pricing Formula

Together with the reference price and the pricing impact values, you can define the pricing formula.

The defined pricing rule now does the following:

- Takes the price of the reference product (e.g. 10 €).
- Looks up the Pricing relevant attributes (ProductLifecycle = growth, PerceivedValue = 70%).
- Converts the attributes to their values  
ProductLifecycle[growth] = 1.2 |  
PerceivedValue[70%] = 2.
- Calculates the price: BasePrice \*  
ProductLifecycle + PerceivedValue = 10 € \*  
1.1 + 2 € = 13 €

#### Advanced Cost

We added support for advanced cost type definition and selection. You can now define different cost types and decide on granular level what cost type should be used in your calculations.

#### Price Setting Package 1.6.0

**Upgrade notes** must be followed for features to work properly. **Standard configuration** is an initial configuration. It does not need to be manually added, but we inform about relevant changes.

#### Bugs

- [\[PFPCS-2212\]](#) - StockCoverDays has error when Forecast Data is null
- [\[PFPCS-2215\]](#) - Unexpected Error displays if Exceptions configuration are empty/no configuration for Exceptions
- [\[PFPCS-2243\]](#) - Unexpected error when Exceptions using PP are overlapped
- [\[PFPCS-2244\]](#) - AdvancedCost- SingleCost that has overlapped validity should return error message
- [\[PFPCS-2251\]](#) - Relevant Competition PP config has error with labels
- [\[PFPCS-2260\]](#) - Stock data do not work when transaction module has failed
- [\[PFPCS-2265\]](#) - ValidTo and ValidUntil in PriceIncrease strategy
- [\[PFPCS-2275\]](#) - Add PromotionPrice strategy to standard config
  - **Standard configuration changes:**
    - New strategy with configuration was introduced. No old engine or strategy has been affected.
      - New entries for PromotionBased strategies in StrategyDefinition
      - New PP called PromotionLookupEngineConfig
- [\[PFPCS-2392\]](#) - Change Label mentions in AnchorEngine
  - **Upgrade note:**
    - In additional configuration tables for Anchor Engine
      - Anchor Label Anchor Field Name
      - Factor To Anchor Field Label Factor To Anchor Field Name
      - Sku Field Label SKU Field Name
- [\[PFPCS-2725\]](#) - Exceptions are ignored and have no affect to FinalListPrice and PriceDecision
- [\[PFPCS-2733\]](#) - Transaction errors happen sometimes when creating new LPG or after recalculating LPG and it will disappear after recalcuting one more time
  - **Upgrade note:**
    - New warnings introduced:
      - TOO\_MANY\_ROWS
      - TOO\_MANY\_ROWS\_ABORTED
- [\[PFPCS-2818\]](#) - Price Insights Dashboard - change format of historical data in 3 static lines

- [PFPCS-2826] - Issue when using P/PXREF as source type in PP AnchorAdditionalConfig
- [PFPCS-2828] - Anchor strategy has issue when Factor To Anchor is null

#### Change Request

- [PFPCS-458] - Value Based Pricing Strategy

#### Tasks

- [PFPCS-1903] - Test modularization
  - [Modularization](#)
- [PFPCS-2192] - Use getConfigEntry for stock data, instead of custom methods for lookups
- [PFPCS-2236] - Add more TDD4C for PriceManager
- [PFPCS-2524] - Add default dependency mapping for Exceptions in StandardConfig
  - **Standard configuration changes:**
    - In DependencyMappingConfig PP
      - Price Exception, -ADJUSTME-, Lookup, -ADJUSTME- Price Exception, DependencyLevelName, Lookup, DependencyLevelName
      - Strategy Exception, -ADJUSTME-, Lookup, -ADJUSTME- Strategy Exception, DependencyLevelName, Lookup, DependencyLevelName
- [PFPCS-2578] - No documentation of validTo

#### Stories

- [PFPCS-660] - Re-fetch PP and PX
  - **Standard configuration changes:**
    - Multiple column labels (translations by database naming) changed
    - Column names (labels by database naming) changed
      - PricingParameter/ExchangeRates/key3  
"Valid Date" -> "ValidDate"
      - PricingParameter/PriceSettingConfig/attribute3  
"Source Table" -> "SourceTable"
      - PricingParameter/PriceSettingConfig/attribute4  
"Source Field" -> "SourceField"
      - PricingParameter/PriceSettingConfig/attribute5  
"Source Field 2" -> "SourceField2"
      - PricingParameter/PriceSettingConfig/attribute6  
"Source Field 3" -> "SourceField3"
      - PricingParameter/PriceSettingConfig/attribute7  
"Source Field 4" -> "SourceField4"
      - PricingParameter/PriceSettingDimensions/attribute1  
"Field Name" -> "FieldName"
      - PricingParameter/PriceSettingLevel/attribute1  
"Price Level" -> "PriceLevel"
      - PricingParameter/PriceSettingLevel/name  
"Price List" -> "PriceList"
      - PricingParameter/StrategyDefinition/attribute1  
"Calculation Engine" -> "CalculationEngine"
      - PricingParameter/StrategyDefinition/attribute2  
"Additional Engine Configuration" -> "AdditionalEngineConfiguration"
      - PricingParameter/StrategyDefinition/attribute3  
"Strategy Calculation Parameters" -> "StrategyCalculationParameters"

- PricingParameter/StrategyDefinition/attribute4  
"Independent Level Only" -> "IndependentLevelOnly"
- PricingParameter/StrategyDefinition/key1  
"Strategy Name" -> "StrategyName"
- ProductExtension/ProductCosts/sku  
"Product Id" -> "Sku"
- ProductExtension/PromotionPrices/sku  
"Part # (sku)" -> "Sku"
- ProductExtension/RecommendedRetailPrices/sku  
"Part # (sku)" -> "Sku"
- [PFPCS-1905] - Attribute based pricing
  - **Upgrade note:**
    - New PPs. They need to be filled in in order to use the Attribute Based engine.
      - AttributeBasedPricingRules
      - IntervalAttributesConversion
      - PricingAttributes
      - ValueAttributesConversion
    - [Attribute Based Engine](#)
- [PFPCS-2234] - Transaction Data lookup overhaul
  - **Upgrade notes (for Transaction module)**
    - New entry in PP PriceSettingConfig. Sample configuration is: Last Period  
Transaction, \*, Months, 3
      - [Last Period Config \(PriceSettingConfig\)](#)
    - If using PX Transaction Source and Price Insight Dashboard, 3 new fields are introduced (Min, Max and Avg).
      - [Transaction Config \(PriceSettingConfig\)](#)
    - New entry in WarningConfig NO\_DATA\_FOR\_LAST\_PERIOD\_LOOKUP

## Price Setting Package 1.5.0

### Bugs

- [PFPCS-1087] - Issue when the name of PP table that is used for Lookup Engine is not correct
- [PFPCS-1940] - Calculation won't work with exceptions module turned off
- [PFPCS-2012] - Error about CalculatedPrices when Net Price module is turned off
- [PFPCS-2312] - Update messages/labels to remove PLCM mentions
  - Change error codes in *WarningConfig* PP:
    - NO\_PLCM\_TARGET->NO\_PF\_TARGET
    - NO\_REASON\_FOR\_PLCM->NO\_REASON\_FOR\_PF
  - Suggestion: For convenient usage of accelerator, adjust business messages to fit your needs.
- [PFPCS-2313] - Raise timeout for our commons lib elements
- [PFPCS-2363] - Can't display Independent Level Decision inside Dependent PL/PG when it depends on Independent XPL
- [PFPCS-2399] - Price Insights Dashboard can't show items from XPG/XPL
- [PFPCS-2420] - MinMarginPrice fails when net price module is turned off
- [PFPCS-2497] - Issue when looking up Actual Price from PX
- [PFPCS-2548] - Unexpected Error about Price Decision displays when Overrides module is turned off
- [PFPCS-2552] - Actual Price lookup from PL only displays after recalculation inside LPG
  - Remove error code from *WarningConfig* PP
    - NO\_RESULT\_PRICE\_IN\_PRICE\_LIST
- [PFPCS-2592] - Changes in standard config for eCommerce
  - If exists, remove *SampleLookupEngineConfig* PP from partition

- Folder name for PSP configuration has changed. Move every PP from “PFXTemplate” to “PFX Accelerator” folder.

#### Stories

- [PFPCS-2235] - Create Standard Configuration for eCommerce
- [PFPCS-2440] - Actual Price Lookup from same LPG (Dynamic Pricing)

### Price Setting Package 1.4.3

#### Bugs

- [PFPCS-2311] - KitEngine and AnchorEngine don't work with Volume Breakdown
- [PFPCS-2355] - Fix PM dependencies for PSP

### Price Setting Package 1.4.2

#### Bugs

- [PFPCS-2245] - Price Selector always has option to override
- [PFPCS-2270] - The price records are getting added to the LPG Global Price Monitor with CostPlus price strategy only, even if the PP table configured with other priceIncrease strategy
- [PFPCS-2272] - Temporary library reference in PriceInsightsDashboard
- [PFPCS-2274] - Fix problems with dependency mapping in Standard Config
  - **Upgrade notes:**
    - Change in RRPLookupEngineConfig PP
      - Key: “Additional Filter Field”
      - Value: “Country” “DependencyLevelName”
    - If standard configuration PXes are used, name for field “Dependency Level Name” has to be changed to “DependencyLevelName” in each of these PXes:
      - DiscountGroups
      - ListPrices
      - ProductCosts
      - RecommendedRetailPrices
      - PromotionPrices

### Price Setting Package 1.4.1

#### Task

- [PFPCS-2257] - Create an upgrade-package definition for PSP in PlatformManager

### Price Setting Package 1.4.0

- [Sub-tasks](#)
- [Bugs](#)
- [Stories](#)
- [Tasks](#)
- [Improvements](#)

#### Sub-tasks

- [PFPCS-1600] - Split PriceSettingConfig into multiple tables
- [PFPCS-1601] - Split CountryMappingConfig into multiple tables
- [PFPCS-1993] - Convert data from different currencies - Actual Price

- [PFPCS-1994] - Convert data from different currencies - Transaction Data - Datamart / Datasource
- [PFPCS-1995] - Convert data from different currencies - Cost
- [PFPCS-2086] - Convert data from different currencies - Transaction Data - PX

## Bugs

- [PFPCS-1604] - Unexpected Error RawMinimumMargin when input wrong type of number into PP MinMargin
- [PFPCS-2003] - Lack of consistency in new Dependency(Level)Mapping
  - **Upgrade notes:**
    - Changed keys in DependencyMappingConfig PP
      - Suffix " Dependency Mapping" has been removed (e.g. "Transaction Dependency Mapping" "Transaction")
    - Changed keys in WarningConfig PP
      - DEPENDENCY\_ADJUSTMENT\_IS\_ZERO  
DEPENDENCY\_LEVEL\_ADJUSTMENT\_IS\_ZERO
      - ERROR\_LOOKING\_UP\_DEPENDENCY\_ADJUSTMENT  
ERROR\_LOOKING\_UP\_DEPENDENCY\_LEVEL\_ADJUSTMENT
      - NO\_DEPENDENCY\_ADJUSTMENT NO\_DEPENDENCY\_LEVEL\_ADJUSTMENT
      - INVALID\_DEPENDENCY\_TABLE\_ID INVALID\_DEPENDENCY\_LEVEL\_TABLE\_ID
- [PFPCS-2004] - No FinalPrice on Independent PLI not handled in Dependent PLI
- [PFPCS-2005] - Change name of key3 in RoundingRulesConfig for Country or CountryName
  - **Upgrade notes:**
    - Changes in RoundingRulesConfig PP:
      - remove label (verify only - should be autodeployed)
      - key3, rename "DependencyMapping" -> "DependencyLevelName"
      - key3, changed translation (label) "Dependency Mapping" -> "Dependency Level Name"
    - DependencyMappingConfig PP (only when using sample data from standard configuration)
      - Actual Price, attr1: "Name" -> "DependencyLevelName"
      - Actual Price, attr3: "Country" -> "DependencyLevelName"
      - Cost, attr1: "Name" -> "DependencyLevelName"
      - Cost, attr3: "Country" -> "DependencyLevelName"
      - Discount Group, attr1: "SalesOrg" -> "DependencyLevelName"
      - Discount Group, attr3: "Country" -> "DependencyLevelName"
      - Price Exception, attr1: "Name" -> "-ADJUSTME-" (Default table does not support dependency mapping)
      - Price Exception, attr3: "Country" -> "-ADJUSTME-" (Default table does not support dependency mapping)
      - Forecast, attr3: "Country " -> "-ADJUSTME" (We don't have a default table for forecasts)
      - Strategy Exception, attr1: "Name" -> "-ADJUSTME-" (Default table does not support dependency mapping)
      - Strategy Exception, attr3: "Country" -> "-ADJUSTME-" (Default table does not support dependency mapping)
      - Forecast, attr3: "Country " -> "-ADJUSTME" (We don't have a default table for transactions)
      - Product Competition, attr3: "country" -> "ADJUSTME-" (We don't have a default configuration for this)
      - Rounding, attr3: "DependencyMapping" -> "DependencyLevelName"
  - [PFPCS-2006] - Wrong config aborts calculation instead of just rounding module
  - [PFPCS-2007] - Remove NO\_ROUNDING\_RULE warning when price is null

- [PFPCS-2008] - Standard config should have different values for rounding rule
- [PFPCS-2013] - Investigate standard config for actualListPrice
- [PFPCS-2014] - Fix standard config in dependencyMappingConfig
- [PFPCS-2016] - Autodeploy does not change column names of PX
- [PFPCS-2017] - Volume breakdowns probably do not work when deploying by PM
- [PFPCS-2018] - Logic to bootstrap parameters won't work properly if there is already parameter with same name and other target date
- [PFPCS-2024] - Strategy with negative result is ignored in Independent PG
- [PFPCS-2026] - Issue when removing duplicates after loading base and standard strategies
- [PFPCS-2027] - Calculation aborted if there is no Currency Conversion
- [PFPCS-2029] - Cost in Independent PG is not affected by different Currency
- [PFPCS-2048] - Hide MinimumMarginPrice warning message when Price Checks module is turned off
- [PFPCS-2055] - Actual Price - Unexpected Error displays when optional column Currency is missing
- [PFPCS-2056] - Lookup ActualPrice from PL - Unexpected Error displays when resultPrice is null
- [PFPCS-2061] - Actual List Price in Dependent PG should lookup resultPrice from Dependent PL
- [PFPCS-2090] - PP LookupEngineConfig doesn't allow to have same SKU but with different DependencyLevelName
- [PFPCS-2092] - Incorrect results return from strategies that are using LookupEngine in DependencyPG
- [PFPCS-2096] - Table dependency mapping do not work on PX lookups which don't utilize "includeEmptyFallback" flag
- [PFPCS-2097] - Table cost lookup does not work
- [PFPCS-2102] - Exceptions do not work with validity period filtering
- [PFPCS-2103] - CalculatedPrices have error in Dependent PG when Independent Level Price is null
- [PFPCS-2108] - Misleading label in PriceSettingDimensions
- [PFPCS-2109] - PLApprovalWorkflow: Execution of ApprovalWorkflow[WorkflowUtils] timed out after 5 units
- [PFPCS-2112] - Incorrect Actual Price is returned in Dependent PG when looking up from PL
- [PFPCS-2113] - Actual Price is looking up from PL without converting Currency
- [PFPCS-2114] - Wrong parameters passed to buildUpToDateFilter()
- [PFPCS-2115] - CalculatedPrices error when Advanced Cost module is turned off
- [PFPCS-2116] - Unable to save ValidFrom/ValidTo in Discount Group PX
- [PFPCS-2117] - Price Exception - Message is not thrown when validity periods of a product are overlapped
- [PFPCS-2118] - AdvancedCost - Unexpected Error Cannot invoke method collectEntries() on null object
- [PFPCS-2120] - Incorrect name of Arithmetic function is using for Advanced Cost - Simple instead of Single
- [PFPCS-2121] - It should throw warning message when no Cost is presented when using Advanced Cost
- [PFPCS-2122] - Change Valid Until to Valid To for all configurable PP/PX lookups
- [PFPCS-2125] - Can't calculate Advanced Cost properly using arithmetic Avg
- [PFPCS-2128] - AvgCost and SumCost should be calculated correctly when setting cost look up with only validFrom
- [PFPCS-2137] - AdvancedCost - Cost cannot be retrieved when AdvancedCost module is off
- [PFPCS-2145] - Valid From and Valid To are not working properly when looking up Exceptions from PP
- [PFPCS-2152] - Discount Group returns null when Valid From and Valid To are not configured in PP PriceSettingConfig
- [PFPCS-2153] - Cannot get property 'PARAM' on null object after PFPCS-2003 was merged
- [PFPCS-2157] - AdvancedCost - Unexpected Error Cannot invoke method collectEntries() on null object when there is no CostSelection user config

- [PFPCS-2164] - Missing ValidFrom/ValidTo feature for PP RoundingRulesConfig lookup
- [PFPCS-2166] - Pricelist: Margin% not taking into account manual override
- [PFPCS-2168] - Some Unexpected Error are displaying after KPI Check was merged
- [PFPCS-2176] - Stock - UNEXPECTED\_ERROR: Unable to convert value [] of type java.lang.String to type net.pricefx.domain.MatrixLookupTableValue3Key
- [PFPCS-2189] - Rounding issue - Calculation Price in Prices matrix should be changed
- [PFPCS-2190] - PriceInsightsDashboard does not have updated lib reference
- [PFPCS-2191] - Add standard configuration for days of cover and adjust it
- [PFPCS-2196] - Transaction Data from DM/DS should apply exchange rates differently
- [PFPCS-2199] - Incorrect type of avgVolumePerDay is using can lead to incorrect Stock Cover Days
- [PFPCS-2201] - Fix Format Type of Stock and Stock Cover Days
- [PFPCS-2202] - Add warning message when Stock / Stock Cover Days null
- [PFPCS-2211] - Error displays when Discount Groups is null
- [PFPCS-2213] - It should ignore the row that has empty DependencyLevelName when looking up Stock data
- [PFPCS-2214] - StockCoverDays can't display value 0
- [PFPCS-2217] - Enhance technical message if there is NO\_SUITABLE\_ROUNDING\_RESULT
- [PFPCS-2218] - Fix small bugs about Labels
- [PFPCS-2222] - Errors about InputDependencyUtil and PriceParameterUtil
- [PFPCS-2228] - CalculatedPrices - Unexpected Error about RoundingRules displays
- [PFPCS-2231] - PriceInsightsDashboard can't display properly
- [PFPCS-2239] - Can't display Cost value if it equals 0
- [PFPCS-2240] - Missing data file for CostTypeDefinitions PP
- [PFPCS-2241] - Missing data file for StockData PP

#### Stories

- [PFPCS-680] - Convert data from different currencies
- [PFPCS-1690] - Price check - Recommended Retail Price
- [PFPCS-1691] - KPI Check - Days of Cover
  - **Upgrade notes:**
    - New entry in PriceSettingConfig:
      - Key: Stock
      - Condition: \* / Independent / Dependent
      - Type: Lookup
      - Source: PX / PP
      - Source Table: name of the table holding data
      - Source Field: name of the column holding stock value
      - Source Field 2 (attribute5): name of the valid from date column. Optional.
      - Source Field 3 (attribute6): name of the valid to the date column. Optional.
    - New entry in DependencyMappingConfig
      - Name: Stock
      - Dependency Field: name of the column in Dependency Configuration PP contains the value to be compared
      - Type: Lookup / Table
      - Mapping Source Field: name of the column in the source table contains the value to be compared
    - New entries in WarningConfig:
      - UNSUPPORTED\_STOCK\_DATA\_SOURCE\_TYPE
      - UNABLE\_TO\_READ\_STOCK\_TABLE
      - NO\_STOCK\_DATA
      - NO\_INPUT\_FOR\_STOCK\_COVER\_DAYS
  - Sample table for data is at the bottom of this page

- [\[PFPCS-1904\]](#) - Multiple lookup sources for Cost data
  - **Upgrade notes:**
    - New module. To turn it on, add a new AP entry
      - key: PSP\_ADVANCED\_COST
      - value: true
    - New Price Parameter CostTypeDefinition. Attributes:
      - name: CostType
      - attribute1: CalculationEngineSuffix
      - attribute2: Type
      - attribute3: CalculationMethod
      - attribute4: SourceTable
      - attribute5: SourceField
      - attribute6: ValidFrom
      - attribute7: ValidTo
      - attribute8: DependencyField
      - attribute9: DependencyType
      - attribute10: MappingSourceField
      - attribute11: Currency
    - New Price Parameter, CostSelection. Attributes:
      - keys are dimensions
      - attribute1: Cost Type #1
      - attribute2: Cost Type #2
      - attribute3: Cost Type #3
      - attribute4: Cost Type #3
      - attribute5: Cost Type #5
    - New entries in WarningConfig:
      - ERROR\_PARSING\_COST\_DEFINITION
      - ERROR\_CALCULATING\_COST
      - ERROR\_LOOKING\_UP\_COST\_SELECTION\_CONFIG
  - Already deployed ProductCosts might be used as a data table

#### Tasks

- [\[PFPCS-2015\]](#) - Enhance technical message in warning about no rounding rules
- [\[PFPCS-2110\]](#) - Write guide to PM deployment
- [\[PFPCS-2198\]](#) - Create autopublishing configuration for PSP
- [\[PFPCS-2219\]](#) - Groovy check instead of null check in rounding utils

#### Improvements

- [\[PFPCS-989\]](#) - BPB should be configured Valid From/ ValidTo same with PLCM
  - **Upgrade notes:**
    - Add ValidFromField and ValidToField configuration in PriceSettingConfig (optional):
      - Actual Price (only works for PX)
      - Cost ("ValidFrom" to attribute6 and "ValidTo" to attribute7)
      - Discount Group
      - Price Exception - Independent
      - Price Exception - Dependent
      - Strategy Exception - Independent
      - Strategy Exception - Dependent
    - In corresponding PP and PX tables fill fields pointed in PriceSettingConfig
      - Those fields should be key(PP)/business key(PX). It is not technical requirement for that but it will be impossible to pass additional parameters otherwise.

- Changes in WarningConfig:
  - PX\_VALIDITY\_PERIODS\_OVERLAPPED
  - PP\_VALIDITY\_PERIODS\_OVERLAPPED
  - DISCOUNT\_GROUP\_FALLBACK\_APPLIED
  - FIELD\_NOT\_FOUND
- This feature is optional, every old table will be working. No new table is needed.



## Price Setting Package 1.3.0

- ✓ There are also videos showing the main new features: [Price Setting Package 1.3.0 Key Features Description](#)

Each of the videos takes less than 3-4 minutes, so the amount of time you need to invest is not that big 😊

- [Notes](#)
  - [Upgrading Projects by PlatformManager](#)
- [Bugs](#)
- [Stories](#)

### Notes

#### *Upgrading Projects by PlatformManager*

There is a known issue: deployment is sometimes aborted when a similar PREF object already exists on the partition. As a workaround, deploy Price Setting Package only on clean partitions or remove the crashing preferences from the partition and try the deployment again.

### Bugs

- [\[PFPCS-1616\]](#) - Color of Corridor columns should overlay color of the row
  - **Upgrade note:** The default value for "NO\_FINAL\_PRICE" record in WarningConfig PP should be changed from "Red" to "Critical"
- [\[PFPCS-1708\]](#) - PB\_ country PL error on line item
- [\[PFPCS-1720\]](#) - Handle unexpected error about Projection and Transaction data when using DM which doesn't exist
  - **Upgrade note:** New entry has been added to WarningConfig PP
    - NO\_SOURCE\_TYPE

- [PFPCS-1730] - Config validation doesn't allow for Linear and LastYear projection config.
- [PFPCS-1737] - YTD data should be counted from the beginning of the current year to the present day
- [PFPCS-1749] - Issues with KitEngine calculation
- [PFPCS-1757] - Handle unexpected errors about Sales Data when Types are not supported
  - **Upgrade note:** New entries have been added to WarningConfig PP
    - NO\_SOURCE\_TABLE
    - NO\_FORECAST\_TYPE
- [PFPCS-1758] - Can't get Sales Data when using Datasource to lookup
- [PFPCS-1775] - Anchor pricing doesn't mark as dirty
- [PFPCS-1776] - Fix reason of Competition strategy in Prices matrix when missing config for Product Competition Country Mapping
- [PFPCS-1779] - Throw error in batchManager # performActionIfNewBatchLoaded
- [PFPCS-1820] - Issues with Global Price/Global Decision when product is only calculated in CountryLPG
- [PFPCS-1821] - Anchor Error - can't return result
- [PFPCS-1827] - Incorrect PriceDecision in case strategy exception is ignored
- [PFPCS-1836] - Base strategies should take places before Standard strategies in Prices matrix
- [PFPCS-1857] - Issues with StrategyException when result of strategy is null or strategy is not in Prices matrix
- [PFPCS-1865] - Fix small issues about formatting
- [PFPCS-1874] - It throws message "Strategy Condition module has failed" if there is no Conditions defined in PP StrategyConditions
  - **Upgrade note:** New entry in WarningConfig PP table:
    - ERROR\_LOOKING\_UP\_STRATEGY\_CONDITIONS\_DATA\_PP
- [PFPCS-1875] - Can't get Override Price when its value equals 0
- [PFPCS-1876] - Price of Strategy(Independent Type Strategy) can't override Price of Exception strategies in FinalListPrice
- [PFPCS-1883] - In case using both price and strategy exceptions, if exception strategy is not in the list of calculated strategies then Price Decision is not correct.
- [PFPCS-1890] - Handle the case when the value is in the Range but can't return the result following the Rule
  - **Upgrade note:** New entry in WarningConfig PP table:
    - NO\_SUITABLE\_ROUNDING\_RESULT
- [PFPCS-1892] - Unexpected Error displays if using \* in Source ID in PP DependencyConfiguration
  - **Upgrade note:** New entry in WarningConfig PP table:
    - INVALID\_DEPENDENCY\_TABLE\_ID
- [PFPCS-1906] - Competition Strategy - Exception is not thrown for dependent LPG
- [PFPCS-1908] - Fix standard config for PSP modules and add trimming values

#### Stories

- [PFPCS-1324] - Make Anchor/Kit Pricing engines input fields parameterized
- [PFPCS-1674] - Definition of relevant Competitors
  - **Upgrade notes:**
    - New entry in WarningConfig PP:
      - ERROR\_LOOKING\_UP\_RELEVANT\_COMPETITION\_DATA\_PP
    - New PP table introduced:
      - Name: RelevantCompetitionData
      - Type/Value: MATRIX/MATRIX3
      - Keys: Product Dimensions (Standard config: Business Unit/Product Group/Product Class)

- Attributes: Relevant Competitor, Competitor#01, Competitor#02, ... , Competitor#29



RelevantCompet...\_ExamplePP.zip

- [\[PFPCS-1711\]](#) - Enhance competition Strategy
- [\[PFPCS-1759\]](#) - Introduce Minimum Margin Price to the Pricing Logic
  - **Upgrade notes:**
    - New entries in WarningConfig PP:
      - ARITHMETIC\_EXCEPTION
      - NO\_MIN\_MARGIN\_PRICE
    - Every configuration row in StrategyDefinition PP that uses CompetitionEngine needs additional input parameter, so COMPETITION\_PRICES must be changed to COMPETITION\_PRICES, MINIMUM\_MARGIN\_PRICE in Strategy Calculation Parameters column.
- [\[PFPCS-1514\]](#) - Price Setting: Global/Local level naming
  - **Upgrade notes:**
    - Change name CountryInformation PP to DependencyConfiguration and do following changes:
      - Change Value Type of this table to MATRIX4
      - Rename fields as follows (label in brackets):
        - key1 DependencyLevelName (Dependency Level Name)
        - key2 DependsOn (Depends On)
        - key3 SourceType (Source Type)
        - key4 SourceId (Source ID)
        - attribute1 Dimension (Dimension)
        - attribute2 Currency (Currency)
        - attribute3 Preference1 (ISO Code)
        - attribute4 Preference2 (SalesOrg)
        - attribute5-30 Preference3-28 (Preference #3-#28) - Only if you intend to use other fields for dependency mapping (old country mapping) purposes
      - Reconfigure dependencies.
        - Old "Global" country should have "Independent" flag in "Depends On" column.
        - Old countries should have "Global" in "Depends On" column
        - Old countries should have PG or PL in "Source Type" column and proper ID of "Global" independent PG/PL
    - Change name CountryMappingConfig PP to DependencyMappingConfig and do following changes:
      - Wherever "Country Field" has Name value, change it to DependencyLevelName

- In "Name" column, change all Country Mapping strings into Dependency Mapping, eg. Cost Country Mapping Cost Dependency Mapping
- Rename fields as follows (label in brackets):
  - attribute1 DependencyField (Dependency Field)
- Rename all autogenerated parameters:
  - \*CountryAdjustment \*DependencyLevelAdjustment
- Rename keys in WarningConfig PP:
  - COUNTRY\_ADJUSTMENT\_IS\_ZERO DEPENDENCY\_ADJUSTMENT\_IS\_ZERO
  - ERROR\_GETTING\_GLOBAL\_PRICES\_FROM\_GLOBAL\_ITEM  
ERROR\_GETTING\_INDEPENDENT\_LEVEL\_PRICES\_FROM\_INDEPENDENT\_LEVEL\_ITEM
  - ERROR\_LOOKING\_UP\_COUNTRY\_ADJUSTMENT  
ERROR\_LOOKING\_UP\_DEPENDENCY\_ADJUSTMENT
  - NO\_COUNTRY NO\_DEPENDENCY\_LEVEL\_NAME
  - NO\_COUNTRY\_ADJUSTMENT NO\_DEPENDENCY\_ADJUSTMENT
  - NO\_GLOBAL\_DECISION NO\_INDEPENDENT\_LEVEL\_DECISION
  - NO\_GLOBAL\_ITEM NO\_INDEPENDENT\_LEVEL\_ITEM
  - NO\_GLOBAL\_PRICE NO\_INDEPENDENT\_LEVEL\_PRICE
  - NO\_INPUT\_FOR\_GLOBAL\_ADJUSTED\_PRICE  
NO\_INPUT\_FOR\_INDEPENDENT\_LEVEL\_ADJUSTED\_PRICE
  - NO\_INPUT\_FOR\_GLOBAL\_PRICE\_PRIORITY  
NO\_INPUT\_FOR\_INDEPENDENT\_LEVEL\_PRICE\_PRIORITY
- Make adjustments in PriceSettingConfig PP:
  - Remove:
    - Row with "Price Orientation" key
    - Row with "Global Price" key
  - Change values:
    - Change all keys:
      - Global Manual Override Allowance Independent Manual Override Allowance
      - Local Manual Override Allowance Dependent Manual Override Allowance
    - Change all Conditions:
      - Global Independent
      - Local Dependent
    - Change all sourceTables if Table type country mapping was used and this placeholder is somewhere in this PP's source table column:
      - <<Country>> <<DependencyPreference>>
- Make adjustments in StrategyDefinition PP:
  - Rename (label in brackets):
    - attribute4 Independent Level Only (Independent Level Only)
    - attribute5 Independent Level Priority (Independent Level Priority)
  - Change values:
    - Global Independent in Level column
    - Local Dependent in Level column
    - all references to COUNTRY\_INFORMATION\_VALUES  
DEPENDENCY\_INFORMATION\_VALUES in  
StrategyCalculationParameters column
  - Change allowed values:
    - Global Independent in Level column
    - Local Dependent in Level column
- Make adjustments in StrategySelection PP:

- Rename:
    - Prioritize Global Price Prioritize Independent Level Price
- Make adjustments in PricingExceptions PP (if used):
  - Rename (label in brackets):
    - key2 -> DependencyLevelName (Dependency Level Name)
- Make adjustments to default PXEs (only if used - labels in brackets):
  - Rename in DiscountGroups PX
    - attribute1 DependencyLevelName (DependencyLevelName)
  - Rename in ListPrices PX
    - attribute1 DependencyLevelName (DependencyLevelName)
  - Rename in ProductCosts PX
    - attribute3 DependencyLevelName (DependencyLevelName)
  - Rename in PromotionPrices PX
    - attribute4 DependencyLevelName (DependencyLevelName)
  - Rename in RecommendedRetailPrices PX
    - attribute4 DependencyLevelName (DependencyLevelName)
- Reconfigure/Recreate all PLS/LPGs that use either GlobalPriceListLogic or CountryPriceListLogic
  - Logic names changed:
    - GlobalPriceListLogic IndependentPriceListLogic
    - CountryPriceListLogic DependentPriceListLogic
  - Gathering input changed, proper values must be selected during first configuration
- Remove unnecessary objects:
  - GlobalPriceListLogic CL
  - CountryPriceListLogic CL
- [\[PFPCS-1545\]](#) - Price Setting: Add conditions to price strategies
  - **Upgrade notes:**
    - New entry in WarningConfig PP:
      - ISSUE\_EXECUTING\_STRATEGY\_CONDITION
    - New PP table introduced:
      - Name: StrategyConditions
      - Type/Value: MATRIX/MATRIX
      - Key: Order
      - Attributes: Condition, Rule, CheckException
      - Example:



StrategyCondi...s\_ExamplePP.zip

- [\[PFPCS-1498\]](#) - Round up / round down feature

- **Upgrade notes:**
  - New entry in DependencyMappingConfig
    - Rounding Dependency Mapping
  - New entry in WarningConfig PP:
    - NO\_ROUNDING\_RULE
  - New module added to AP:
    - PSP\_ROUNDING\_RULES\_MODULE
  - New entry in PriceSettingConfig, standard config below:
    - Key: Rounding Rules
    - Condition: \*
    - Type: Lookup
    - Source: PP
    - Source Table: RoundingRule
    - Source Field: RoundingRule
    - Source Field 2 : RoundingMode
  - New PP table introduced:
    - Name: RoundingRulesConfig
    - Type/Value: MATRIX/MATRIX2 (when using Table Dependency Mapping), MATRIX /MATRIX3 (when using Lookup Dependency Mapping)
    - Keys: From, To, Dependency Mapping (in case of Lookup)
    - Attributes: Rounding Rule, Rounding Mode
    - Warning: Empty RoundingRulesConfig is currently not supported. To turn off rounding, the module should be turned off or one entry with the rounding mode "NO\_ROUNDING" should be created.
    - Example:



RoundingRulesC...\_ExamplePP.zip

- [\[PFPCS-1515\]](#) - Enable to create Qty Breakdown Pricelist
  - **Upgrade notes:**
    - New entries in WarningConfigPP:
      - ERROR\_LOOKING\_UP\_VOLUME\_EXCEPTION
      - ERROR\_PARSING\_VOLUME\_DISCOUNTS
      - NO\_INPUT\_FOR\_VOLUME\_DISCOUNT
    - New PP tables introduced:
      - VolumeBreakdown
        - Name: {DependencyLevelName}VolumeBreakdown
        - Type/Value: MATRIX/MATRIX3
        - Keys: Product Dimensions (Standard config: Business Unit/Product Group /Product Class)

- Attributes: Volume #01, Discount #01, Volume #02, Discount #02, ... , Volume #15, Discount #15
- VolumeExceptions
  - Name: VolumeBreakdownExceptions
  - Type/Value: MATRIX/MATRIX2
  - Keys: SKU, Dependency Level Name
  - Attributes: Volume #01, Discount #01, Volume #02, Discount #02, ... , Volume #15, Discount #15
- Examples:



- [\[PFPCS-1165\]](#) - Wrap pricing logic inside manager - make price calculations centralized
  - **Upgrade notes:**
    - Entries removed from WarningConfig PP:
      - ERROR\_LOOKING\_UP\_BASE\_STRATEGIES
      - ERROR\_LOOKING\_UP\_PRICE\_STRATEGIES
      - NO\_INPUT\_FOR\_COUNTRY\_ADJUSTED\_PRICE
      - NO\_INPUT\_FOR\_GLOBAL\_PRICE
      - NO\_INPUT\_FOR\_GLOBAL\_DECISION
      - NO\_PRICE\_IN\_PRICE\_SELECTOR
      - NO\_STRATEGY\_DEFINITION\_PP
      - NO\_STRATEGY\_IN\_PRICE\_SELECTOR
      - NO\_PX\_ROWS\_FOR\_PRICE\_LOOKUP
    - New entries added to WarningConfig PP:
      - ERROR\_LOOKING\_UP\_STRATEGIES
- [\[PFPCS-1698\]](#) - Rename Projection to Forecast in configuration
  - **Upgrade notes:**
    - Changes in PriceSettingConfig
      - "Projection" keys are changed to "Forecast"
    - Changes in DependencyMappingConfig
      - "Projection Dependency Mapping" changed to "Forecast Dependency Mapping"
- [\[PFPCS-1703\]](#) - PM/PSP v1.3.0: Move all elements into definition directory

### Price Setting Package 1.3.0 Key Features Description

- [Volume Breakdown](#)
- [Conditions in Strategy Selection](#)
- [Global/Local Handling Refined to "Independent/Dependent"](#)
- [Rounding Rules for Prices](#)

## Volume Breakdown

You can now create a Matrix PL/LPG based on Volume/Discount Levels. Volume and Discount are configured via a PP per the Lookup Key Level. It is also possible to add exceptions on the SKU level.

## Conditions in Strategy Selection

You can create conditions that have influence on execution of strategies. You can skip, fallback and re-prioritize strategies based on simple rules.

## Global/Local Handling Refined to "Independent/Dependent"

The original approach was to handle Global and Country Price Lists. Now we made this mechanism more generic to take care of various business requirements of independent / dependent Price Lists to cover more than simple "country" relations.

## Rounding Rules for Prices

In the new release we have added a rounding mechanism. You can pick from different rounding rules based on value ranges.

## Price Setting Package 1.2.1

### Bugs

- [\[PFPCS-974\]](#) - Hide Changes element in PL
- [\[PFPCS-1659\]](#) - ActualListPrice - Unexpected Error disappears after recalculating inside LPG
  - **Upgrade note:** Change in WarningConfig, removed NO\_PX\_ROWS\_FOR\_PRICE\_LOOKUP entry
- [\[PFPCS-1680\]](#) - TransactionData error is displayed when working on Price Insights Dashboard
- [\[PFPCS-1695\]](#) - Incorrect handling of NullPointerException
- [\[PFPCS-1696\]](#) - Forecast data lookups should be done using current-year date filter
- [\[PFPCS-1697\]](#) - Fix standard config to be up-to date
- [\[PFPCS-1704\]](#) - YTD lookup for transaction data has no upper bound

### Tasks

- [\[PFPCS-1702\]](#) - PM/PSP v1.2.1: Move all elements into definition directory

## Price Setting Package 1.2.0

- [Notes](#)
  - [Upgrading Projects by PM](#)
  - [Lightning Project](#)
- [Sub-tasks](#)
- [Bugs](#)
- [Stories](#)
- [Tasks](#)
- [Improvements](#)

### Notes

#### *Upgrading Projects by PM*

There is a known issue: deployment is sometimes aborted when a similar PREF object already exists on the partition. As a workaround, deploy PSP only on clean partitions or remove the crashing preferences from the partition and try deployment again.

## Lightning Project

On the Lightning partition, no preferences are imported. These are changes in preferences which may be adjusted manually:

- Added preferences for:
  1. AvgCompetitionAdditionalConfig
  2. CostPlusAdditionalConfig
  3. MaxCompetitionAdditionalConfig
  4. MinCompetitionAdditionalConfig
  5. PriceIncreaseAdditionalConfig
- Changed preference for:
  1. ProductCosts PX - added visibility for CountryName column

## Sub-tasks

- [PFPCS-470] - Add error handling when config is incorrect ( ex. util returning null when we want get label)
- [PFPCS-1110] - Setup for "required" user-configuration
- [PFPCS-1185] - Configurable "Abort" flag on warning codes
- [PFPCS-1186] - Show warnings matrix popup on aborted calculation
- [PFPCS-1217] - Define modules that the package will be split into
- [PFPCS-1218] - Implement module (de)activation mechanism
- [PFPCS-1219] - Define requirements for PM to facilitate modular deployment

## Bugs

- [PFPCS-762] - Syntax Error warning when calculating TransactionData and ForecastData with incorrect Source/Source Table
- [PFPCS-947] - Error when AdditionalEngineConfiguration and StrategyCalculationParameters are missing in PPStrategyDefinition
- [PFPCS-1035] - Overrides syntax error when no Price Exception Country Mapping and Strategy Exception Country Mapping are defined
- [PFPCS-1179] - Error when Exchange Rate is 0 or is not defined
- [PFPCS-1241] - Unexpected Error GlobalCalculatedItem is displays when missing Global Price source table
- [PFPCS-1281] - Unexpected Error UsedListPrice displays when Manual Override Allowance is not configured
- [PFPCS-1302] - Use const config when lookuping MinMargin
- [PFPCS-1352] - Error displays when PP PriceSettingDimensions is not configured
- [PFPCS-1362] - Remove GlobalMinPrice from global pricelist
- [PFPCS-1364] - Check if AllowStrategyOverride is obsolete
- [PFPCS-1372] - Remove obsolete UserSelectedPrice
- [PFPCS-1373] - Syntax error on setting country to WarningManager
- [PFPCS-1473] - Handle exception with no message
- [PFPCS-1477] - Incorrect counter on Warnings manager
- [PFPCS-1528] - Fix warning Unexpected Error for element RawCompetitionData
- [PFPCS-1530] - Counter on Prices popup should start at 1 not 0
- [PFPCS-1536] - Minimum Margin can't be displayed if it is greater than Margin
- [PFPCS-1538] - Unexpected Error about PriceChangeEffect because warning message is not deployed properly
- [PFPCS-1541] - Price Setting: PL - Products don't use correct strategy
- [PFPCS-1542] - PB: PL - strategy is not updated with too many products
- [PFPCS-1548] - Fix Format Type for PriceChangeEffect

- [PFPCS-1551] - Handle warning message of Price Change Effect in case Price Level is Gross
- [PFPCS-1553] - Unexpected Error appears when the product is only calculated in CountryLPG
- [PFPCS-1555] - Can't get Actual Price from PL
- [PFPCS-1556] - Actual Price = 0 from PX List Prices is ignored
- [PFPCS-1566] - Messages in Technical messages column are not consistent
- [PFPCS-1569] - Missing Technical message for BaseStrategies in GlobalLPG
- [PFPCS-1572] - Incorrect message displaying when both PP tables about Corridor are not presented
- [PFPCS-1577] - Error displays when calculating HeadquarterPLG without Country defined in Price Orientation
- [PFPCS-1579] - Unexpected Error GlobalAdjustedPrice displays if missing PP()CountryAdjustment
- [PFPCS-1580] - Adjust the Price Insights Dashboard to new Higcharts library
- [PFPCS-1582] - Fix configuration for ActualListPrice
- [PFPCS-1585] - Displaying error ChartBuilder Error(@0):null for some products
- [PFPCS-1586] - Error Execution of PriceInsightsDashboardLogic[Util] timed out after 5 units
- [PFPCS-1595] - Can't lookup RRP price from source type PP
- [PFPCS-1596] - Lookup Engine: Cannot invoke method getConfigEntries() on null object
- [PFPCS-1597] - KitEngine returns OK status when no BoM data is present
- [PFPCS-1606] - Error displays if no strategies in PriceSelector
- [PFPCS-1609] - Missing library in LookupEngine and AnchorEngine
- [PFPCS-1610] - Message TOO\_SMALL\_MARGIN should not be displayed if Margin null
- [PFPCS-1612] - Missing 3 lines Min,Max,Avg on Price Insights Dashboard
- [PFPCS-1630] - Missing Preferences for tables
- [PFPCS-1631] - Error when accessing TransactionConfig in PriceInsightsDashboard
- [PFPCS-1632] - Should hide ConfigManager columns from LPG
- [PFPCS-1634] - Unexpected Error about Global Price displays after modularization
- [PFPCS-1635] - Error Net Price module has failed
- [PFPCS-1636] - Error Overrides Module has failed - Unexpected Error about PriceSelector
- [PFPCS-1643] - Transaction module has failed - Unexpected Error about PriceChangeEffect
- [PFPCS-1653] - Can't return value when LookupEngine is using in BaseStrategySelection and StrategySelection for the same product
- [PFPCS-1656] - Remove Business Key from PX Product Cost that is used for Lightning
- [PFPCS-1659] - ActualListPrice - Unexpected Error disappears after recalculating inside LPG
  - **Upgrade note:** Change of error code in WarningConfig:
    - NO\_PX\_ROW\_FOR\_PRICE\_LOOKUP NO\_PX\_ROWS\_FOR\_PRICE\_LOOKUP
- [PFPCS-1660] - Incorrect config for NetEngine (TargetPrice)

#### Stories

- [PFPCS-271] - Simulation Effects of Price Change
  - **Upgrade note:** New key in WarningConfig:
    - NO\_INPUT\_FOR\_PRICE\_CHANGE\_EFFECT
- [PFPCS-990] - Handle country mapping in competition data
  - **Upgrade note:** New entry in PP CountryMapping called "Product Competition Country Mapping". Warning: PComp tables do not have metadata, hence attributeId must be supplied in attribute3's column (usually for PCOMP it will be "country")
- [PFPCS-1057] - Enhancements for PriceIncreaseEngine
  - **Upgrade note:** There are some changes to configuration:
    - Name changed - In *StrategyDefinition* you have to change *PriceIncreaseEngine* to *AdjustmentEngine*
    - Configuration added - To keep the same config as in v1.1, use *CostPlusAdditionalConfig* and *PriceIncreaseAdditionalConfig* in *Additional Engine Configuration* column of *StrategyDefinition*.
- [PFPCS-1184] - Manage aborting calculation

- **Upgrade note:** New keys in PP WarningConfig:
  - EMPTY\_CONFIG
  - NO\_CONFIG
- **Upgrade note:** Removed keys in PP WarningConfig:
  - NO\_INPUT\_FOR\_OVERRIDE\_REMOVER
  - NO\_EXCEPTION\_CONFIG
  - NO\_TRANSACTION\_DATA
  - NO\_TRANSACTION\_CONFIG
  - NO\_STRATEGIES\_DEFINITION
  - NO\_PRICE\_LOOKUP\_CONFIG
  - NO\_PRICE\_LEVEL
  - NO\_ORIENTATION\_CONFIG
  - NO\_INPUT\_FOR\_PRICE\_SELECTOR
  - NO\_INPUT\_FOR\_MANUAL\_PRICE\_REASON
  - NO\_INPUT\_FOR\_GET\_DECISION
  - NO\_INPUT\_FOR\_EXCHANGE\_RATE
  - NO\_INPUT\_FOR\_EXCEPTIONS
  - NO\_INPUT\_FOR\_ALLOW\_MANUAL\_STRATEGY\_OVERRIDE
  - NO\_INPUT\_FOR\_ALLOW\_MANUAL\_PRICE\_OVERRIDE
  - NO\_GLOBAL\_ITEM\_LOOKUP\_CONFIG
  - NO\_GLOBAL\_ITEM\_FOR\_GLOBAL\_PRICE
  - NO\_GLOBAL\_CURRENCY
  - NO\_FORECAST\_DATA
  - NO\_FORECAST\_CONFIG
  - NO\_EXCEPTION\_ALLOWANCE\_CONFIG
  - NO\_DISCOUNT\_GROUP\_CONFIG
  - NO\_DIMENSIONS
  - NO\_CURRENCY
  - NO\_COUNTRY\_MAPPING\_FOR\_TRANSACTION
  - NO\_COUNTRY\_MAPPING\_FOR\_STRATEGY\_EXCEPTION
  - NO\_COUNTRY\_MAPPING\_FOR\_PRICE\_LOOKUP
  - NO\_COUNTRY\_MAPPING\_FOR\_PRICE\_EXCEPTION
  - NO\_COUNTRY\_MAPPING\_FOR\_FORECAST
  - NO\_COUNTRY\_MAPPING\_FOR\_DISCOUNT\_GROUP
  - NO\_COUNTRY\_MAPPING\_FOR\_COST
  - NO\_COUNTRY\_LOOKUP\_CONFIG
  - NO\_COUNTRY\_LOOKUP
  - NO\_COST\_CONFIG
  - CANT\_FETCH\_TRANSACTION\_DATA
  - CANT\_FETCH\_PRICE\_ORIENTATION\_CONFIG
  - CANT\_FETCH\_FORECAST\_DATA
  - CANT\_CONVERT\_CURRENCY
- [\[PFPCS-1216\]](#) - Modularization of Price Setting Accelerator
- [\[PFPCS-1539\]](#) - Prepare post-deployment instruction for replacing "Global->Country" naming to something else

#### Tasks

- [\[PFPCS-741\]](#) - Scripts for platform manager (BPB)
- [\[PFPCS-1239\]](#) - Adjustment of warning manager to new validation methods
- [\[PFPCS-1367\]](#) - Remove submodules from Price Setting Accelerator
- [\[PFPCS-1435\]](#) - Remove api.retainGlobal
- [\[PFPCS-1436\]](#) - Prepare release notes for Price Setting Package v1.2

- [PFPCS-1465] - Split MinimumMargin into 2 elements
- [PFPCS-1508] - Create cost data PX for Lightning
- [PFPCS-1574] - Remove empty Library element from GlobalPriceList logic
- [PFPCS-1576] - Don't use api.getElement("WarningManager") inside Commons lib
- [PFPCS-1657] - Add module configuration AP to repository

#### Improvements

- [PFPCS-973] - Add priority # to prices pop-up
- [PFPCS-1503] - Adjust Competition Pricing Strategies in Sample Config
- [PFPCS-1529] - Add "TechnicalMessage" column to warning matrix
- [PFPCS-1651] - Do not return strings when module/calculation has been aborted

### Price Setting Package 1.1.1

#### Bugs

- [PFPCS-1474] - Fix standard configuration for PSP v1.1

### Price Setting Package 1.1.0

- [Sub-tasks](#)
- [Bugs](#)
- [Tasks](#)
- [Improvements](#)

#### Sub-tasks

- [PFPCS-326] - Merge PriceSettingLevel into CountryInformation
- [PFPCS-490] - Use Manhattan feature instead of our own implementation for mapping metadata with attribute ids
- [PFPCS-868] - ConfigUtils.getLookupKeys() should be reworked to use caching - big gain
  - **Upgrade note:** "INVALID\_DIMENSIONS" - Entry with that error code should be added to WarningConfig
- [PFPCS-869] - Use closures for CalculatedPrices data initialization - big gain
- [PFPCS-871] - ActualPriceLookup should be reworked to use caching - big gain
- [PFPCS-872] - Add batching of competition data lookups - medium gain
- [PFPCS-873] - Competition lookup is executed twice - medium gain
- [PFPCS-874] - Country element lookups should be cached - medium gain
- [PFPCS-937] - PM script testing
- [PFPCS-1062] - Change names of repositories

#### Bugs

- [PFPCS-766] - Fix Reason displayed inside Prices matrix
- [PFPCS-882] - Product Currency column should be removed from PL/PG
  - **Upgrade note:** Product Currency column should be manually hidden from PL/PG
- [PFPCS-982] - Can't get Sales Data from Datamart
- [PFPCS-983] - Message when SKU is not defined in Source for Anchor
- [PFPCS-1036] - Incorrect Reason when using Price Exception and Global/Local Manual Override "Yes"
- [PFPCS-1037] - Issues about chart feature and rounding number on Price Insight Dashboard
- [PFPCS-1038] - TransactionData do not work after changing name of the element from Configuration to ConfigManager
- [PFPCS-1083] - Misleading counter on warnings manager

- [PFPCS-1107] - Make changes in code appropriate to new name of element of PopUpData , RawPrices
- [PFPCS-1125] - Missing COUNTRY\_INFORMATION\_CONFIG key in ConfigManager
- [PFPCS-1133] - Issue UNEXPECTED\_ERROR LookupKeys
- [PFPCS-1134] - We cannot use custom calculation engines anymore
- [PFPCS-1150] - Unexpected Error about ForecastData displays in Warnings matrix
- [PFPCS-1155] - Incorrect FinalListPrice when setting "ExceptionTable" in Global/Local Manual Override Allowance
- [PFPCS-1163] - Change config of PX List Prices
- [PFPCS-1164] - Can't get Global Price and Global Decision from GlobalLPG
- [PFPCS-1167] - Exception not caught in warning manager
  - **Upgrade note:** "NO\_EXCEPTION\_CONFIG" - Entry with that error code should be added to WarningConfig
- [PFPCS-1171] - Unexpected Error about ActualPriceLookup appears after full recalculation but disappears after recalculating inside LPG
- [PFPCS-1172] - Can't get Forecast Data from Datamart
- [PFPCS-1192] - Should throw warning message when Actual List Price is null in PX List Prices
  - **Upgrade note:** "NO\_ACTUAL\_LIST\_PRICE" - Entry with that error code should be added to WarningConfig
- [PFPCS-1193] - Issue with Price Insight Dashboard after some changes in ConfigManager
- [PFPCS-1195] - TransactionData cannot be read, unexpected error occurs
- [PFPCS-1201] - Should change the Reason of Competition strategy that is displayed in Prices matrix when no Competition Data
- [PFPCS-1226] - Misleading message in lookupEngine error
- [PFPCS-1242] - Fix error handling while loading exceptions data from price parameter
- [PFPCS-1249] - AdditionalCalculatorParameters should not be visible
- [PFPCS-1251] - Fix country mapping in PriceException
- [PFPCS-1252] - Incorrect Kit price and Anchor price when using PriceSetting Level "Gross"
- [PFPCS-1257] - Unexpected Error displays when PPStrategySelection is not presented
  - **Upgrade note:** There was error in WarningConfig, there was two same error codes, called ERROR\_LOOKING\_UP\_STRATEGIES, one of them should be renamed to ERROR\_LOOKING\_UP\_PRICE\_STRATEGIES
- [PFPCS-1263] - Error occur when choosing Price Insights Dashboard
- [PFPCS-1269] - Price strategy that has value as 0 is ignored when type of Manual Override Allowance is "No"
- [PFPCS-1271] - Manual Price Reason is not displayed when using Price exception with type of Manual Override Allowance is "Yes"
  - Behaviour of exceptions and column names has been changes. We strongly suggest to create new PL/PG instead of recalculating old ones
  - **Upgrade note:** "NO\_INPUT\_FOR\_MANUAL\_PRICE" - Entry with that error code should be removed from WarningConfig
- [PFPCS-1275] - Incorrect Price Decision in case Strategy Exception is ignored
- [PFPCS-1278] - Incorrect warning message displays when Price Override is not allowed
- [PFPCS-1280] - Fix warning message about Override Remover
  - **Upgrade note:** "NO\_INPUT\_FOR\_OVERRIDE\_REMOVER" - Entry with that error code should be added to WarningConfig
- [PFPCS-1288] - Should throw warning for Margin when FinalPrice is null or FinalPrice is 0
- [PFPCS-1291] - Missing warning message when Min Margin is null
- [PFPCS-1328] - Fix Kit Engine error message when Sub-component or Raw Material is null
- [PFPCS-1330] - PriceSelector should be hidden when Manual Override Allowance is "No" or "ExceptionTable"
- [PFPCS-1337] - Fix Exceptions Manager

- [PFPCS-1341] - HQ LPG should get the warning message that is defined for Country
  - **Upgrade note:** Manual entries in WarningConfig are working fine for global country now. It can change behaviour, as previously it was ignored
- [PFPCS-1365] - When Discount % is null then if FinalListPrice = 0 it will not be used in other calculations
- [PFPCS-1369] - Non consistent preferences
- [PFPCS-1428] - Country should be mandatory field, not just throw exception when it's null
  - Country for CountryPriceListLogic is mandatory field now, calculation won't be saved without filling that value

#### Tasks

- [PFPCS-812] - Change names of configs about price/strategy overrides
  - **Upgrade note:** In PP PriceSettingConfig:
    - Keys with "Global Manual Override" should be renamed to: "Global Manual Override Allowance"
    - Keys with "Local Manual Override" should be renamed to: "Local Manual Override Allowance"
- [PFPCS-813] - Change globalOrientation Configs
- [PFPCS-817] - Remove "Product" prefix from BPB PXes
- [PFPCS-884] - Verify quality of Preferences and Labels on clean deployment
  - **Upgrade note:** Multiple labels and preferences has been changed. Each config should be revised if it is displayed properly
- [PFPCS-885] - Move TransactionDataUtil from lib to Common
- [PFPCS-900] - Change naming of lookupKeys
- [PFPCS-904] - Remove getLookupKeys method from configManager
- [PFPCS-933] - Remove SKU from Anchor engine config, and change engine code according to that
- [PFPCS-980] - Change manager calls to be less repetitive/more informative in Price Builder
- [PFPCS-996] - Prepare logics to work with debugger
- [PFPCS-1008] - Do not return transaction data and forecast data as element result
- [PFPCS-1010] - Maintain not crashing debugger for BPB v1.1
- [PFPCS-1039] - Change placement of the method
- [PFPCS-1060] - Add groovydoc to utility elements
- [PFPCS-1073] - Fix Sonar bugs and smells for Price Builder Accelerator
- [PFPCS-1082] - Remove fork of Shared lib from Library
  - **Upgrade note:** Most recent SharedLib should be deployed
- [PFPCS-1089] - Move exchange rates element out of Library
- [PFPCS-1090] - Add better separation for external inputs to PriceCalculator
  - **Upgrade note:** Implementation of custom engines has been changed, new parameters for Calculator Engines should be added from scratch.
- [PFPCS-1093] - Remove existing PriceLevel config from PriceSettingConfig
  - **Upgrade note:** In PP PriceSettingConfig, key2 for "Global Price" entry should be renamed to "\*"
- [PFPCS-1106] - NetPriceLevel should use ConfigManager and return boolean value
- [PFPCS-1127] - Unify usage of const config in config manager
- [PFPCS-1140] - Clean up exception and override handling
- [PFPCS-1158] - Failing TDD4C after manual conflict resolution
- [PFPCS-1166] - Unify names of fields in const config

#### Improvements

- [PFPCS-1385] - Make Country mandatory for Country Price List Logic