Accelerate Price Setting Package

Version 1.7.1

April 2021

Accelerate Price Setting Package 1.7.1

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

- Price Setting Business Introduction 1.7.1
- Price Setting Overview 1.7.1
- Price Setting Configuration 1.7.1
- Price Setting Package Administration Procedures 1.7.1
- Release Notes PSP 1.7.1

Price Setting Business Introduction 1.7.1

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.
- 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.
- 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
- 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
    - You can have **sales and forecast data** displayed.
    - You can create custom price behavior by creating **exceptions** and allowing **manual overrides.**
    - 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 Overview 1.7.1

Price Setting Use Cases 1.7.1

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

- **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.

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

- For Standalone Price List scenario you can set up more dependencies, but there will be no link between these dependencies and they will share the same parameters.
- 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.

Price Setting Concepts 1.7.1

It is important to understand general concepts of the package to be able to use it correctly and become fluent with various package configurations.

The Price Setting Package works with the following general concepts:

- Product Segmentation 1.7.1
- Strategy Importance Configuration 1.7.1
- Hierarchy 1.7.1
- Modularization 1.7.1
- Conditions 1.7.1
- Warning Handling 1.7.1

Price Strategies 1.7.1

The Price Setting Package provides several built-in price strategies; they are described in detail in Calculation Engines 1.7.1:

- Adjustment Engine 1.7.1
- Attribute Based Engine 1.7.1
- Anchor Engine 1.7.1
- Competition Engine 1.7.1
- Kit Engine 1.7.1
- Lookup Engine 1.7.1
- Net Engine 1.7.1
- Custom Engines 1.7.1

You can also add your own price strategies.

Product Segmentation 1.7.1

All parameters can be configured across segments of the product portfolio. During deployment and setup, you can choose up to 5 product attributes as "dimensions" to differentiate all the setup parameters. This way you can choose different pricing strategies, surcharges, etc. depending on different segments of your product portfolio.

Strategy Importance Configuration 1.7.1

Every strategy has to be properly defined. There are the following configuration options:

- '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

The order works differently with independent and dependent price lists:

- Independent price list calculates:
    1. Base strategies
    2. Standard strategies
    3. Removes base strategies which returned no price.

- Dependent price list is more complex, since it has several configuration options which tweak the strategies order:
    1. 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 PP. In this case, "Independent Level Adjusted Price" will be put at the end of strategies. It is a configuration on the product level.
    2. 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.
    3. Prices from the independent price list can be ignored on the dependent level by setting "Independent Level Only" to "Yes" in the Strategy Definition PP.
    4. Dependent prices come in pairs with independent level adjusted prices (with dependent before independent), unless "Independent Level Priority" is set. 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.

Hierarchy 1.7.1

Calculation logics contain the final output values. One of them is a price decision (which contains info about which price / price strategy we calculate, or reason for a price), and the other is Final Price which contains price after calculations. Exceptions and overrides have a strong connection to these values.

With the default settings, our hierarchy looks like this:

Order	Name	Price to calculation	Price Decision	Necessary Action
1	Manual Price Override	Price from the Manual Price field	Default comment is inserted if none given. It can be manually overridden.	Type a price in the Manual Price 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	Base strategy from PriceSelector	Price from the first price strategy	Name of the price strategy.	N/A

We can skip some levels of this hierarchy by changing Manual Override Allowance configuration. For example, when we set the Independent Manual Override for a Price to "ExceptionTable", it will disable the Manual Price Override from this hierarchy.

Modularization 1.7.1

Price Setting Package is split into multiple modules. Each of them turns on and off one of the features.

Available modules:

Module Name	Description	Advanced Property Name
Transaction	Displays transaction and forecast data about products. Stock data is independent from transactions, but calculation of StockCoverDays is dependent on this module.	PSP_TRANSACTION_MODULE
Net Price	Allows to calculate a net price (with a proper discount taken into consideration).	PSP_NET_PRICE_MODULE
Exceptions	Handles exceptions in pricing. It allows to manually override product prices on the PGI or PP configuration (per SKU) level.	PSP_OVERRIDES_MODULE
Price Checks 1.7.1	Checks if the user margin is within a suitable range and if not, 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 Price Flexibility Package. 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.	PSP_PRODUCT_COMPETITION_MODULE
Strategy Condition	Performs additional checks if prices meet certain conditions.	PSP_STRATEGY_CONDITION_MODULE
Rounding Rules	Rounds prices to user friendly values.	PSP_ROUNDING_RULES_MODULE
Advanced Cost 1.7.1	Calculates additional cost types.	PSP_ADVANCED_COST

To enable a module, a corresponding Advanced Property must be set to "true" on a partition. Moreover, every config needed for the module must be deployed.

Each of the modules is fairly independent. Most of them have warnings on the module level, so in case something goes wrong in the module, the rest of them will still work.

Conditions 1.7.1

The 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 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.

There is a predefined Price Parameter called StrategyConditions. To learn about its structure, see Strategy Conditions.

In this section:

- Conditions

- Order
- Wildcards

Conditions

We support evaluating prices as shown below:

- Gross price of any calculated strategy or exception. The syntax is: "xxx.price"
- Margin of any calculated strategy or exception. The syntax is: "xxx.margin"
- Any value supplied to PriceCalculator as a parameter. Also custom values from additionalParameters and additionalOptionalParameters are accepted.

Supported operators are: "<", ">", "=".

The left-hand side of an equation must always be a property of a given strategy. The right-hand side of an equation may be either a property of a strategy or a PriceCalculator's parameter. The suggested usage of this feature is to think of left-hand side strategy as the main strategy for the given condition.

In addition, the right-hand side may also be modified by a certain fraction. To do that, you need to wrap the right-hand side into parenthesis "(...)" and add an asterisk character, followed by a string parsable to BigDecimal. E.g. "Cost+.price<(RRP.price * 2)"

Please keep in mind that the strategy name needs to be written explicitly. This is the case of Dependent Price List. "Cost+ (Independent Type Strategy)" and "Cost+" are two different strategies.

Order

There are three options to change the price order:

- Removing a strategy put on the left-hand side. The syntax is: "(skip)"
- Moving a strategy in the left-hand side to the last position of the strategy list. The syntax is: "(fallback)"
- Moving a strategy after another strategy. The syntax is: "$strategyName1<$strategyName2". Example: "Cost+<RRP"

Wildcards

If you manage a large number of strategies in one go, you can use a wildcard: "{any}". Each rule with such a wildcard will be multiplied by every strategy and exception. At the place of a wildcard, strategyName will be inserted. E.g. "{any}.margin<MINIMUM_MARGIN"

Warning Handling 1.7.1

- Warning Codes List
- Warning Placement
- Warning Configuration

Warning Codes List

The 'WarningConfig' Price Parameter contains the following codes:

- ARITHMETIC_EXCEPTION
- CANT_GET_REASON
- CANT_READ_DATA_FOR_PRICE_LOOKUP

- CANT_READ_DISCOUNT
- DEPENDENCY_LEVEL_ADJUSTMENT_IS_ZERO
- EMPTY_CONFIG
- ERROR_GETTING_INDEPENDENT_LEVEL_PRICES_FROM_INDEPENDENT_LEVEL_ITEM
- ERROR_LOOKING_UP_CORRIDOR_CONFIG
- ERROR_LOOKING_UP_DEPENDENCY_LEVEL_ADJUSTMENT
- ERROR_LOOKING_UP_MINIMUM_MARGIN
- ERROR_LOOKING_UP_RELEVANT_COMPETITION_DATA_PP
- ERROR_LOOKING_UP_STRATEGIES
- ERROR_LOOKING_UP_VOLUME_EXCEPTION
- ERROR_PARSING_VOLUME_DISCOUNTS
- EXCEPTION_IGNORED
- EXCEPTION_STRATEGY_OVERRIDDEN
- INVALID_DIMENSIONS
- INVALID_DEPENDENCY_LEVEL_TABLE_ID
- ISSUE_EXECUTING_STRATEGY_CONDITION
- NO_ACTUAL_LIST_PRICE
- NO_CONFIG
- NO_CORRIDOR_CONFIG
- NO_COST_ENTRY_IN_COST_TABLE
- NO_DEPENDENCY_LEVEL_ADJUSTMENT
- NO_DEPENDENCY_LEVEL_NAME
- NO_DISCOUNT_VALUE
- NO_EXCHANGE_RATE
- NO_FINAL_PRICE
- NO_FORECAST_TYPE
- NO_INDEPENDENT_LEVEL_DECISION
- NO_INDEPENDENT_LEVEL_ITEM
- NO_INDEPENDENT_LEVEL_PRICE
- NO_INPUT_FOR_CORRIDOR
- NO_INPUT_FOR_DISCOUNT
- NO_INPUT_FOR_INDEPENDENT_LEVEL_ADJUSTED_PRICE
- NO_INPUT_FOR_INDEPENDENT_LEVEL_PRICE_PRIORITY
- NO_INPUT_FOR_MARGIN
- NO_INPUT_FOR_MIN_MARGIN_HANDLING
- NO_INPUT_FOR_NET_PRICE
- NO_INPUT_FOR_PRICE_CHANGE_EFFECT
- NO_INPUT_FOR_VOLUME_DISCOUNT
- NO_MIN_MARGIN
- NO_MIN_MARGIN_PRICE
- NO_PL_FOR_PRICE_LOOKUP
- NO_PF_TARGET
- NO_REASON_FOR_PF

- NO_ROUNDING_RULE
- NO_SALES_VOLUME_FORECAST
- NO_SALES_VOLUME_LAST_YEAR
- NO_SALES_VOLUME_YTD
- NO_SOURCE_TABLE
- NO_SOURCE_TYPE
- NO_STRATEGY_DEFINITION_ENTRY
- NO_STRATEGY_OVERRIDE_ALLOWED
- NO_SUPPORTED_TABLE_TYPE
- NO_TURNOVER_FORECAST
- NO_TURNOVER_LAST_YEAR
- NO_TURNOVER_YTD
- TOO_SMALL_MARGIN
- UNABLE_TO_READ_COST_TABLE
- UNEXPECTED_ERROR
- UNEXPECTED_PRICE_RANGE_FOR_CORRIDOR
- UNSUPPORTED_EXCEPTION_TYPE
- UNABLE_TO_READ_STOCK_TABLE
- PP_VALIDITY_PERIODS_INVALID_FORMAT
- NO_STOCK_DATA
- NO_INPUT_FOR_STOCK_COVER_DAYS
- UNSUPPORTED_ACTUAL_PRICE_SOURCE_TYPE
- TOO_MANY_ROWS
- TOO_MUCH_ROWS_ABORTED

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 signalizes 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 details on setup see WarningConfig PP 1.7.1.

Price Setting Features 1.7.1

Unlike price setting concepts, you don't have to know all features to be able to use the package. These are all separate functionalities that can be skipped.

The following functionality of the Price Setting Package can be enabled:

- Sales and Forecast Data 1.7.1
- Exceptions and Manual Overrides 1.7.1
- Price Checks 1.7.1
- Rounding 1.7.1
- Advanced Cost 1.7.1
- Volume Breakdown 1.7.1
- Price Insights Dashboard 1.7.1

Sales and Forecast Data 1.7.1

Sales and forecast data can be displayed by the Transaction module.

When transaction data lookups are configured, the following information is shown:

- Historical Data
  - Last Year Sales Volume
  - Last Year Turnover
  - Sales Volume YTD
  - Turnover YTD
  - Last Period Turnover
  - Last Period Volume
- Forecast Data
  - Sales Volume Forecast
  - Turnover Forecast

Notes

Exchange Rate

Transaction data may point to Product Extensions, Datamart or Data Source. Only difference between DM and DS is that prices in DM are already converted to DM's currency. Even when DM's entry has the "currency" field, this field should be ignored (as exchange rate has been applied during transition of data from DS to DM). The exchange rate is applied between DM's currency and calculation's currency only if they differ.

Row Limits

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 don't expect to have tens of thousands rows. We expect data to be pre-aggregated. Having a lot of rows in PX is possible, but counter intuitive. 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.

Time Periods

The last period configuration allows for flexible data lookup. Introduced time units (days, weeks, months and years) are not just for user experience – 1 week is not equal to 7 days. Calculation takes into consideration only periods which are finished. According to the Pricefx calendar implementation, a week starts on Sunday and ends on Saturday.

Example: lookup of data on 22 September 2020 with the configuration "1 week" will use data from 13 September to 19 September.

For details on lookup and configuration options see Price Setting Configuration Price Parameters 1.7.1.

Exceptions and Manual Overrides 1.7.1

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

Exceptions

Depending on the configuration, you can allow for exceptions:

- on a line level in Price List / LPG;
- through a specific exception table;
- or both.

Manual Overrides

In configuration, you can display these elements in used Price Lists / LPGs:

- Manual Price / Manual Price Decision – Fields that allow you to enter a price and comment manually. This price will be used as a final used price with the comment as the price decision.
- PriceSelector – Drop-down field where you can choose one of already calculated prices.

Configuration

Exceptions and overrides can be configured separately on a dependent and independent level. For details on configuration see Price Setting Configuration Price Parameters 1.7.1.

Price Checks 1.7.1

The Price Checks module verifies if the user margin is within a suitable range and if not, issues alerts. Also, for dependent price lists, it checks if the difference between a dependent and independent price is within a suitable range.

Price checks are performed on calculated results. Values for price check results are configurable in respective pricing parameters.

Price Check	Independent Level	Dependent Level	Description
	Yes	Yes	Minimum margin for a specific product checked against the calculated margin.

Minimum Margin			
Adjusted Price Corridor	No	Yes	Deviation between the independent level adjusted price and final list price.
List Price Corridor	No	Yes	Deviation between the independent level price and final list price.
Minimum Margin Price	Yes	Yes	A price calculated based on available minimum margin percent, cost and discount data. This is a minimum price at which Minimum Margin will be achieved. Formulas used for calculation are: 1. Gross calculation: MinMarginPrice = Cost / (1 - MinMargin%) 2. Net calculation: MinMarginPrice = Cost / (1 - Discount% - MinMargin% + MinMargin% * Discount%)

Rounding 1.7.1

The Rounding module allows users to round prices to business friendly values. In this Accelerator only the **List Price** is rounded for each calculated strategy. It means that when using the Net Calculation Level, only the Gross Price will be rounded.

Rounding is done by using a defined set of rules. This set can be expanded if needed.

The module comes with some predefined rules:

- To49Cents (xxx.49)
- To50Cents (xxx.50)
- To95Cents (xxx.95)
- To99Cents ((xxx.99)
- ToWhole (xxx.00)
- To5Whole (xx5.00)
- To49Whole (x49.00)
- To99Whole (x99.00)
- RawPrice (xxx.xx)
- NoRounding (xxx.xxxxxxxx)

⚠ Please note that "cents" does not suggest currency. It refers to rounding to decimal places.

Advanced Cost 1.7.1

The Advanced Cost module allows users to choose among different types of cost calculations. Currently, three types are supported:

- SINGLE - Simple lookup for only one cost.
- AVG - Calculates the average cost out of every cost in the targeted Product Extension.
- SUM - Calculates the sum of costs in the targeted Product Extension.

Similarly to strategies, every lookup may have different cost types selected. All of them will be calculated and presented. Only the first of them will be used for margin calculation. Calculation strategies might use each of the calculated cost.

While defining the types of costs, a proper suffix is defined. The base name of the engine parameter is always "COST". If the cost type has the "_EXAMPLE" suffix, then you can access that parameter in StrategyDefinition by COST_EXAMPLE. The first calculated cost has an additional alias "PRODUCT_COST" which makes switching the module off/on easier.

Each of the cost definitions has its own dependency mapping config and PX configuration (including optional valid dates or currency). They are all defined inline.

By enabling the AdvancedCost module, Cost configuration in PriceSettingConfig and DependencyMappingConfig becomes unused and can be removed.

For configuration details see CostTypeDefinition.

Volume Breakdown 1.7.1

It is possible to run the PL/LPG calculation with a volume breakdown. This allows you to apply different volume discounts, depending on the item quantity.

Quantities and discounts are looked up with product dimensions in one of {$DependencyLevelName} VolumeBreakdown parameters, but exceptions on the SKU level are also supported (for details see Volume BreakdownExceptions PP 1.7.1 configuration).

When using this feature, a calculation item with a volume equaling to 1 will be added. It will keep the default price without the volume discount and will be used as an independent item price in dependent calculations.

This feature needs some additional configuration, as described in VolumeBreakdownExceptions.

Price Insights Dashboard 1.7.1

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.

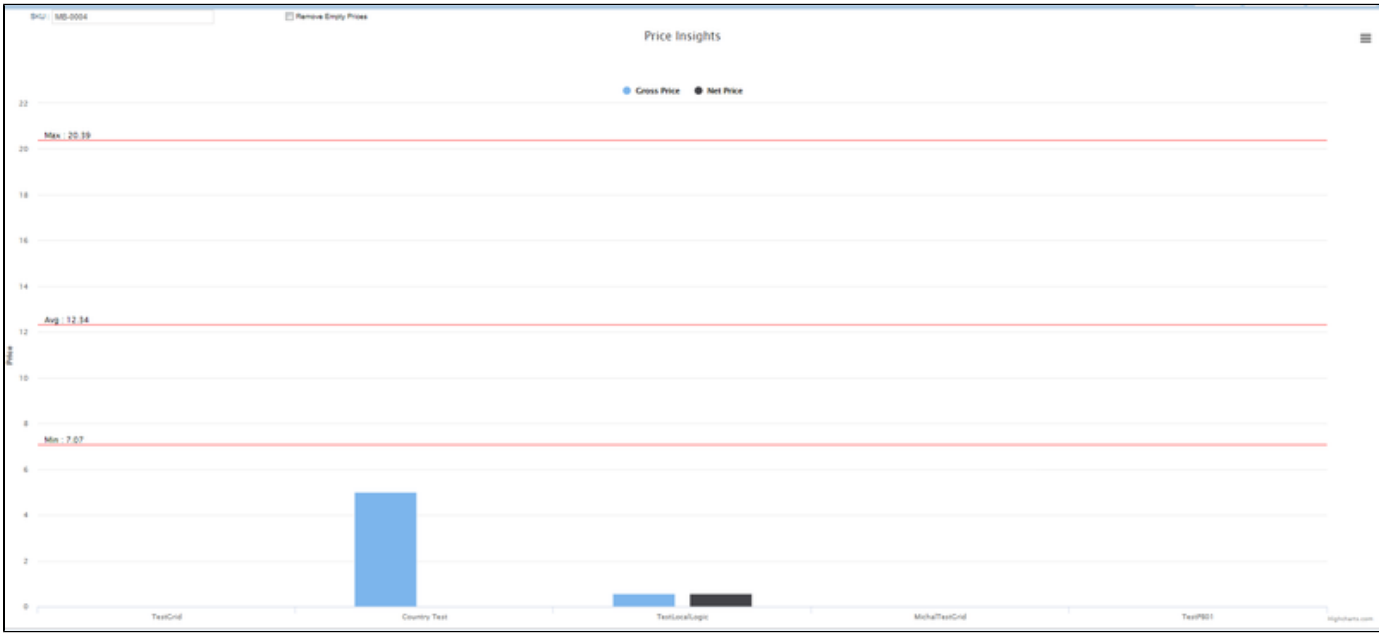
- User Inputs
- Analysis
- Data Requirements and Deployment

User Inputs

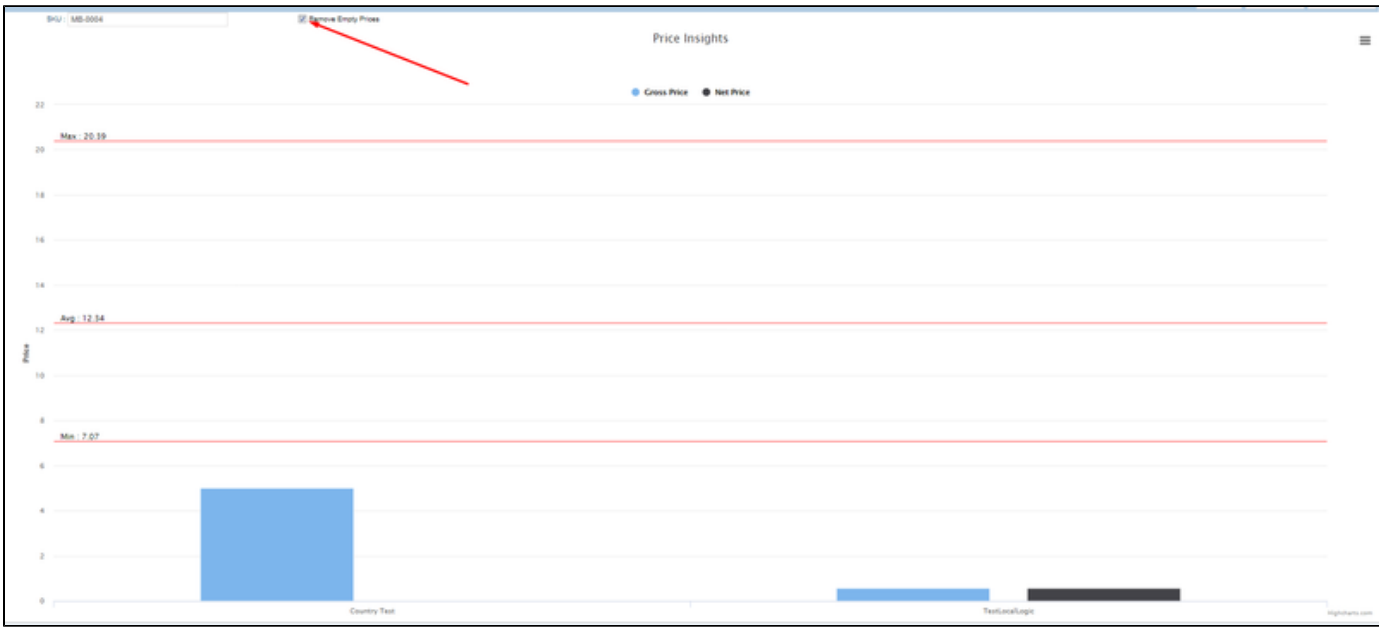
- **SKU** - SKU of the product you want to show in the dashboard.
- **Remove Empty Prices** - Defines if price lists and price grids with empty prices should be filtered out.
- **Independent Level Name** - Choose an independent level that you want to show. All price lists and price grids with the selected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

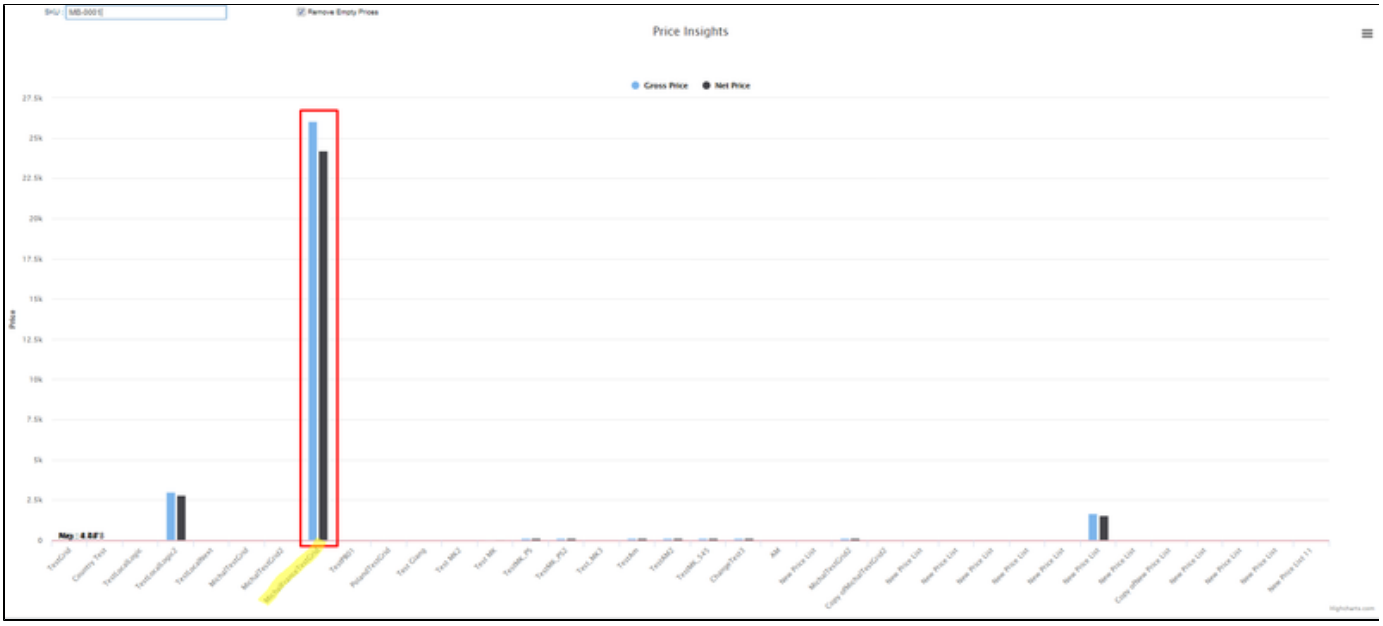


This dashboards presents data correlated with SKU MB-0004. There are three price lists / price grids with empty results, so this is something to be checked. You can filter out these empty results using the checkbox.



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 data	Historical data about transactions	Yes
Configuration in PP	Contains configuration for transaction Data Source	Yes

Calculation Engines 1.7.1

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 the engine's detailed documentation.

All available strategies are configured using the StrategyDefinition Price Parameter. More information about it can be found in Configuration Price Parameters and in an individual engine's documentation.

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
Adjustment Engine	Takes one price as a base and applies a factor to it.	• Cost Plus

		• Price Increase
Anchor Engine	Calculates prices based on the price of another SKU. Note: This engine is deprecated.	• Anchor Pricing
https://pricefx.atlassian.net/wiki/spaces/ACCDEV/pages/2517468375/Attribute+Engine	Calculates prices based on the price of another SKU and current SKU's attributes impact value.	• Anchor Pricing
Competition Engine	Calculates prices according to existing competition prices.	• Competition Based Pricing
Kit Engine	Calculates prices of a kit based on subcomponent prices.	• Kit Pricing
Lookup Engine	Looks up prices from an existing table.	• RRP • Promotion Pricing • Everything that works with a lookup of a stored price
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 LookupEngine.	• Net Pricing
Custom Engines	Any custom library method can be used as an engine as long as it takes proper parameters and returns a proper result.	• Basically anything

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 1.7.1

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: `Value + Adjustment`
	"Percentage"	Adjustment is a percentage. Result is: `Value * (1 + Adjustment)`
	"SellingPrice"	Adjustment is a percentage. Result is: `Value / (1 - Adjustment)`

Default Strategy Calculation Parameters

For Cost+: `PRODUCT_COST`, `PLUS_FOR_PRODUCT`
For Price Increase: `BASE_PRICE`, `PRICE_INCREASE`

Attribute Based Engine 1.7.1

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:

- 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: ba5ac71a-26b9-45e9-9e54-67872052f444#media-blob-url=true&id=8f37fd0a-4f82-44b2-8ff0-d22b7b7ac8a7&collection=&contextId=87696&mimeType=image%2Fpng&name=image-20200819-072649.png&size=7700&width=870&height=68
- 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.

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: • PX • PP • P • PXREF	Defines where the anchor data is kept.
Source Table	ExampleTableName	Name of the data table. Expected only when PX or PP *Source Type* 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
```

Define Attribute Data (with Sample Data)

AttributeBasedPricingRules PP

Name	Operator #1	Pricing Attribute #1	Operator #...	Pricing Attribute #...
Attribute-Based Simple	+	Color	+	Size
...	...	...	/	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 1.7.1

⚠ 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: • PX • PP • P • PXREF	Defines where the anchor data is kept.
Source Table	ExampleTableName	Name of the data table. Expected only when PX or PP *Source Type* 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 1.7.1

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: `TargetCompetitorPosition = NumberOfCompetitors * Percentage`
- Price Position - Select a price at the given percentage point. The formula for calculation is: `Price = (CompetitorMaxPrice + CompetitorMinPrice) * Percentage`

Important note: Competitor Position and Price Position cannot be used at the same time.

After the price has been selected, 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.

Input Parameters

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

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: • `min` - Selects the competitor with the lowest price. • `max` - Select the competitor with the highest price. • `min + X` - Selects the competitor with the lowest price and adjusts the position by X. • `max – X` - Selects the competitor with the highest price and adjusts the position by X. • `40%` - Selects the competitor whose	Selects the target competitor to compare to. Case insensitive.

	position is at 40th percentile of the whole competitor range.	
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: • Yes • No	Checks whether the selected competitor price is affordable based on Minimum Margin Price. If it is not, the next competitor 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"	• yes - relevant competitors are defined • no - the list of competitors is excluded
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

```
COMPETITION_PRICES (or RELEVANT_COMPETITION_DATA),MINIMUM_MARGIN_PRICE
```

Kit Engine 1.7.1

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 `api.bomList()` 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 1.7.1

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.

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: • PX • PP	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.
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 1.7.1

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: • PX • PP	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.
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 matches some data passed in the Additional Filter Values list.

Default Strategy Calculation Parameters

`SKU,TARGET_DATE,DISCOUNTS,DEPENDENCY_INFORMATION_VALUES`

Custom Engines 1.7.1

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

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

This method 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.

Price Setting Configuration 1.7.1

This section is intended for Configuration Engineers to guide them through package deployment and features setup and to provide technical documentation.

- Price Setting Deployment 1.7.1
- Price Setting Configuration Price Parameters 1.7.1
- Price Setting Advanced Technical Information 1.7.1

Price Setting Deployment 1.7.1

The easiest 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.

When it comes to providing CSV data on dependency level configuration and for PriceSettingDimensions, we suggest to have attributeIds in the first line of the CSV file as headers, so that PlatformManager can automatically map it with the proper fields. Otherwise, manual mapping might be required.

Here are samples:

Accelerators deployed through PlatformManager do not set default business keys on PX/CX tables. Thus, any integration using PlatformManager and the CSV via SFTP template will, out of the box, be inserts rather than upserts.

Price Setting Configuration Price Parameters 1.7.1

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

- Actual Price Config (PriceSettingConfig) 1.7.1
- CompetitionAdditionalConfig PP 1.7.1
- Cost Config (PriceSettingConfig) 1.7.1
- CostTypeDefinition PP 1.7.1
- Dependency Mapping Config (PriceSettingConfig) 1.7.1
- DependencyConfiguration PP 1.7.1

- Discount Group Config (PriceSettingConfig) 1.7.1
- Exception Config (PriceSettingConfig) 1.7.1
- Exceptions and Manual Override Allowance Config (PriceSettingConfig) 1.7.1
- ExchangeRates PP 1.7.1
- Forecast Config (PriceSettingConfig) 1.7.1
- Last Period Config (PriceSettingConfig) 1.7.1
- PriceSettingDimensions PP 1.7.1
- PriceSettingLevel PP 1.7.1
- PricingExceptions PP 1.7.1
- Rounding Rules Config (PriceSettingConfig) 1.7.1
- Stock Config (PriceSettingConfig) 1.7.1
- StrategyConditions PP 1.7.1
- StrategyDefinition PP 1.7.1
- Transaction Config (PriceSettingConfig) 1.7.1
- VolumeBreakdownExceptions PP 1.7.1
- WarningConfig PP 1.7.1

Actual Price Config (PriceSettingConfig) 1.7.1

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	• PX • PL • PG	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.

CompetitionAdditionalConfig PP 1.7.1

Column name	Name	Mode
Values	{Setting Name}	{Setting Value}
Description	To learn about the built-in pricing engines, see Calculation Engines 1.7.1.	

⚠ 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:

Price Parameter Values : CompetitionAdditionalConfig [5]

Name	Mode
Competitor Position	min
Repositioning %	
Force Margin Check	no
Repositioning Abs	+10
Price Position	

Cost Config (PriceSettingConfig) 1.7.1

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.
Condition	• Independent • Dependent • *	• Independent - Used on an independent price list. • Dependent - Used on a dependent price lists. • *{} - Used for both independent and dependent price lists. It also overrides any existing configuration with "Independent " or "Dependent" conditions.
Type	Lookup	

Source	PX	Currently, only a PX lookup is supported.
Source Table	{PX name}	Remember to use "name" from the configuration, not a label.
Source Field	{name of the column with the product cost}	
Source Field 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.
Source Field 3	{name of the column with the start date of the validity period, typically ValidFrom}	Used for searching with a validity period. 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. When the ValidFrom field is set but the ValidTo field is left empty, it will find the value with the closest validity date. 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.
Source Field 4	{name of the column with the end date of the validity period, typically ValidTo}	

CostTypeDefinition PP 1.7.1

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	"Look up"	• SINGLE • AVG • SUM	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 DependencyConfiguration (country mapping feature)	• Table • Lookup (country mapping feature)	Name of the PX column (country mapping feature)	Name of the PX column with currency code
Description	Name of the definition,	String concatenated with "COST"	Other types will be adde	For details see Advanced Cost 1.7.1.	Note: Name is required							

	used in cost selection.	and passed to engines as a parameter.	d in the future.		here, not a label.							

Dependency Mapping Config (PriceSettingConfig) 1.7.1

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

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 and it directly impacts configuration found in PriceSettingConfig. Entries in this table have to be present even if you intend to use only the independent level price list.

Column	Value	Description
Key	• Cost • Discount Group • Transaction • Projection • Price Exception • Strategy Exception • Actual Price • Product Competition • Rounding • Stock	Defines how to filter out values from different dependencies. There must be one row for every key defined in this table.
Dep	{name of the label in	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

endencyField	the DependencyConfiguration PP}	column configured in Source Field of this configuration row. In the start it has the "~~*ADJUSTME*~~" value and it must be changed. Special use cases: • 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.
Type	• Lookup • Table	We can do the dependency mapping mechanism in two ways: • **Lookup** – Assumes that you have 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. • **Table** – 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 which will be swapped with our dependency property defined by the dependency mapping mechanism. The placeholder has this format: <<DependencyPreference>>. In this type of value, the Mapping Source field is ignored because you do not need to filter results inside the data source. An example: • Dependency Level Name: Germany • Preference1 (from Dependency Configuration): DE • Dependency Field: Preference1 • Source Type: PX • Source Table (from PriceSettingConfig): Product Costs <<DependencyPreference>> With this data, the dependency mapping mechanism will search for a PX named Product Costs DE and then perform the lookup.
MappingSourceField	{name of the column with the dependency information in the source data}	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("${Source Field}", "${valueRetrieviedFromCondition}") 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".

DependencyConfiguration PP 1.7.1

Column Name	Dependency Level Name	Depends On	Source Type	Source ID	Dimension	Currency	Preference #01-28
Values	{name of dependency}	{independent level name}	{type of source data for independent lookup}	{ID of the source data for independent lookup}	This is arbitrary metadata used in the table to help describe what the dependency is used for. This value is not used in any calculation.	{currency code}	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 table. Currently, only one field can be used to map at a time.
Description	e.g. "Germany"	e.g. "Global"	Allowed values: • PG • XPG • PL • XPL	e.g. "155"	e.g. "Country", "Warehouse"	e.g. "EUR"	e.g. "ISO Code", "SalesOrg"

Example:

Price Parameter Values : DependencyConfiguration [9]

Dependency Level Name	Depends On	Source Type	Source ID	Dimension	Currency	ISO Code	SalesOrg
Asia	Independent	*	*		EUR	GL	SO20
Germany	Global	PG	289	Country	EUR	DE	SO31
Global	Independent	*	*		EUR	GL	SO20
MainDistribution	Independent	*	*		EUR		
Stationaryshop	MainDistribution	*	*	Channel	EUR		
Warehouse1	Asia	*	*	Warehouse	EUR	DE	SO31
Warehouse2	Asia	*	*	Warehouse	EUR	DE	SO31
Webshop	MainDistribution	*	*	Channel	EUR		
Japan	Global	PG	289	Country	EUR	JP	SO31

Discount Group Config (PriceSettingConfig) 1.7.1

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

Column	Value	Description
Key	Discount Group	Defines where data about discount groups are stored.
	*	Condition is not needed here. Leave it at the default value.

Condition		
Type	Lookup	Hardcoded value.
Source	PX	Hardcoded value. For now, only PX is supported.
Source Table	{name of PX}	Remember to use "name" from the configuration, not a label.
Source Field	{name of the column with the product group}	Must be filled when the price grid was created.
Source Field 2	{name of the column with the start date of the validity period, typically ValidFrom}	Used for searching with a validity period. 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. When the ValidFrom field is set but the ValidTo field is left empty, it will find the value with the closest validity date. 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.
Source Field 3	{name of the column with the end date of the validity period, typically ValidTo}	

Exception Config (PriceSettingConfig) 1.7.1

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	• Strategy Exception • Price Exception	Holds the exception configuration. There is one row for each type.
Cond	• Independent • Dependent	• Independent – Used for an independent calculation. • Dependent – Used for a dependency calculation (any other dependency level name).

it i on		
T y pe	Lookup	Hardcoded value.
S o u r ce	• PX • PP	Type of the source table.
S o u r c e T a b le	{name of PX or PP}	Remember to use "name" from the configuration, not a label. ⚠ For PP lookups, we assume two things: 1. The only existing key is the SKU code, dependency mapping is not applied. 2. If there are two keys: a. key1 – SKU code b. key2 – Dependency mapping value
S o u r c e F i e ld	{name of the column with the exception data for a product}	The type of this column depends on the exception type. • Strategy – String with the name of an existing strategy definition. **Importa nt note**: 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. • Price – Numeric value with a price override.
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}	Used for searching with a validity period. 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. When the ValidFrom field is set but the ValidTo field is left empty, it will find the value with the closest validity date. 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.
S o u r	{name of the column with the	

ceField3	end date of the validity period, typically ValidTo}	

Exceptions and Manual Override Allowance Config (PriceSettingConfig) 1.7.1

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	• Independent Manual Override Allowance • Dependent Manual Override Allowance	Holds the configuration of exception/override allowance. One row should be configured for every entry.
Condition	• Strategy • Price	Defines which exception type is configured. There is one row for each condition.
Type	• Yes • No • LineLevel • ExceptionTable	Defines if and what kind of exception is allowed: • 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. • ExceptionTable - Only exceptions through the exception tables are available (defined in ExceptionConfig). It also means that line level manual overrides in elements "ManualPrice", "ManualPriceReason" and "PriceSelector" are cleared. • Yes - Both exception types are allowed. The importance hierarchy is as follows: 1. Price Manual Override 2. Strategy Manual Override 3. Price Exception 4. Strategy Exception • No - Exceptions are not allowed. Elements "ManualPrice", "ManualPriceReason" and "PriceSelector" are hidden from the user.

ExchangeRates PP 1.7.1

	From	To	ValidDate	Rate

Column name				
Values	{code of the base currency matching the DependencyConfiguration data}	{code of the target currency matching the DependencyConfiguration data}	{from which date the currency is valid}	{exchange rate value}
Description	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

Forecast Config (PriceSettingConfig) 1.7.1

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	• Q1 • Q2 • Q3 • Q4	Logic checks the current date. If it matches this configuration row, the row is used.
Type	Last Year	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	• Q1 • Q2 • Q3 • Q4	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

Column	Value	Description
Key	Forecast	Defines how to calculate the forecast.
Condition	• Q1 • Q2 • Q3 • Q4	Logic checks the current date. If it matches this configuration row, the row is used.
Type	Lookup	Logic will perform a lookup for the forecast from an external source.
Source	• Datamart • Datasource • PX	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

Last Period Config (PriceSettingConfig) 1.7.1

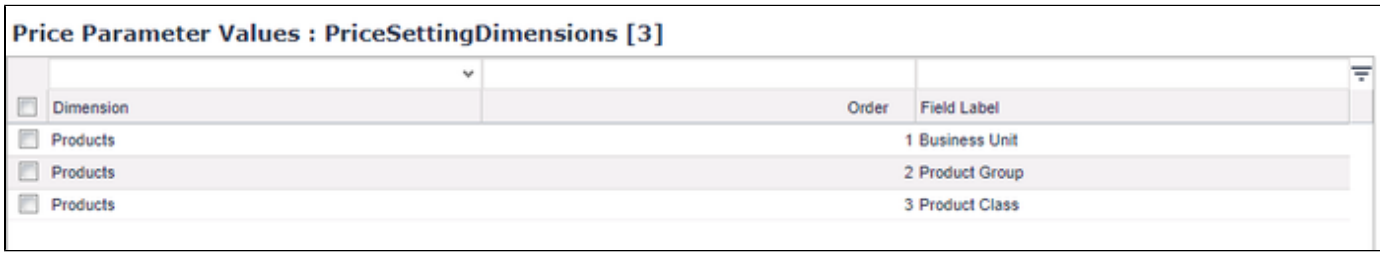
This configuration represents a single row of the PriceSettingConfig Price Parameter. It is not related to lookup sources. It is a candidate to be moved to the module-specific table. For now, ignore column names in that config.

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	• Years • Months • Weeks • Days	Time unit of lookup configuration
attribute2	{any integer}	Amount of time units of lookup configuration

PriceSettingDimensions PP 1.7.1

Column name	Dimension	Order	Field Name
Values	Products	• 1 • 2 • 3 • 4 • 5	{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.	

Example:

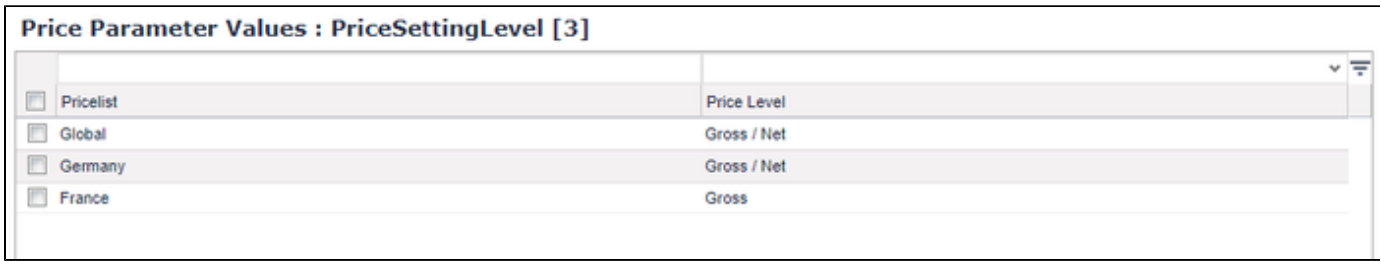


PriceSettingLevel PP 1.7.1

Column name	Pricelist	Price level

Values	• {name of dependency level } as stated in DependencyConfiguration PP	• Gross • Gross / Net
Description		If an additional discount from a gross to net price should be calculated for a specific dependency level.

Example:



PricingExceptions PP 1.7.1

Column name	**SKU**	**Dependent Level Name**	**Price Exception**	**Strategy Exception**
Values	{SKU from Product master}	{name of dependency level} as stated in DependencyConfiguration PP	Defined price exception for given parameters	Defined strategy (as stated in StrategyDefinition PP) exception for given parameters
Description	e.g. "MB-0001"	e.g. "Global"	e.g. "20.0"	e.g. "Cost+"

Example:

Rounding Rules Config (PriceSettingConfig) 1.7.1

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.

Stock Config (PriceSettingConfig) 1.7.1

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	• Independent • Dependent • *	• Independent – Used on an independent price list. • Dependent – Used on a dependent price list. • *{} – Used for both independent and dependent price lists. It also overrides any existing configuration with "Independent " or "Dependent" conditions.
Type	Lookup	
Source	• PX • PP	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: 1. When dependency mapping is not applied, there is only one key which is the SKU code 2. For other cases: a. key1 – SKU code 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		Used for searching with a validity period. When the ValidFrom field (Source Field 2) is left empty, the

Field 2	{name of the column with the start date of the validity period, typically ValidFrom}	ValidTo field (Source Field 3) will be ignored and the normal search approach will be applied. When the ValidFrom field is set but the ValidTo field is left empty, it will find the value with the closest validity date. 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.
Source Field 3	{name of the column with the end date of the validity period, typically ValidTo}	

StrategyConditions PP 1.7.1

For additional details see also Conditions 1.7.1.

Column name	Order	Condition	Rule	CheckException
Values	{incrementing integer}	{condition}	{rule}	• Yes • No
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 Conditions 1.7.1.	Defines what to do with the strategy when the condition is evaluated to true. For details see Order.	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.

StrategyDefinition PP 1.7.1

Column name	Strategy Name	Level	Calculation Engine	Additional Engine Configuration	StrategyCalculationParameters	Independent Level Only	Independent Level Priority	Overridable
Values	{user friendly name of the str	• Independent • Dependent	• {name of the predefined engine} or • {path	{name of PP with the engines customization}	{list of parameter names which are sent to calculation}	• Yes • No	• Yes • No	• Yes • No

	at egy}		to the function in groovyLibrary which will perform calculations} To learn about the built-in pricing engines, see Price Strategies 1.7.1.					
Description	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.	Default parameters are: • SKU • TARGET_DATE • PRODUCT_COST • BASE_PRICE • LOOKUP_KEYS • PLUS_FOR_PRODUCT • COMPETITION_PRICES • MINIMUM_MARGIN_PRICE • PRICE_INCREASE • DEPENDENCY_INFORMATION_VALUES • BOM_LIST • DISCOUNTS If additional parameters are necessary, they need to be added from the	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.

					code to the CalculatedPrices element in the "additionalParameters" map. The parameters are passed as an input to engines, so the order is important and should not be changed for default engines.			

Example:

Price Parameter Values : StrategyDefinition [18]

Strategy Name	Level	Calculation Engine	Additional Engine Configuration	Strategy Calculation Parameters	Independent Level Only	Independent Level Priority
RRP	Independent	LookupEngine	RRPLookupEngineConfig	SKU,TARGET_DATE,COUNTRY_INFORMATION_VALUES		
RRP	Dependent	LookupEngine	RRPLookupEngineConfig	SKU,TARGET_DATE,COUNTRY_INFORMATION_VALUES		
PriceIncrease	Independent	AdjustmentEngine	PriceIncreaseAdditionalConfig	BASE_PRICE,PRICE_INCREASE		
PriceIncrease	Dependent	AdjustmentEngine	PriceIncreaseAdditionalConfig	BASE_PRICE,PRICE_INCREASE		
MinCompetition	Independent	CompetitionEngine	MinCompetitionAdditionalConfig	COMPETITION_PRICES		
MinCompetition	Dependent	CompetitionEngine	MinCompetitionAdditionalConfig	COMPETITION_PRICES		
MaxCompetition	Independent	CompetitionEngine	MaxCompetitionAdditionalConfig	COMPETITION_PRICES		

Transaction Config (PriceSettingConfig) 1.7.1

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	• Datamart • Datasource • PX	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.

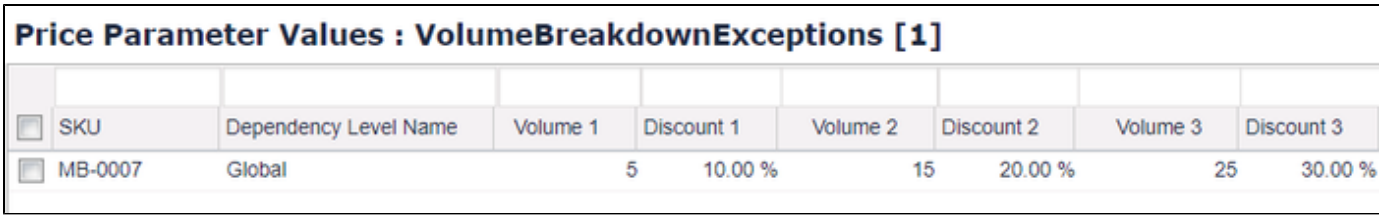
Source Field 4	{name of the column with the date for the invoice}	⚠ The column must be of the Date type.
Source Field 5	{name of the column with the currency}	Invoice price currency
Source Field 6	{name of the column with minimum value of aggregated data}	Only for pre-aggregated PX Source
Source Field 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

VolumeBreakdownExceptions PP 1.7.1

For additional details see also Volume Breakdown 1.7.1.

Column name	SKU	Dependent Level Name	Volume #01...#15	Discount #01...#15
Values	{sku from Product master}	{name of dependency level} as stated in DependencyConfiguration PP	Value representing the product volume that will use the given discount	Discount for the given volume
Description	e.g. "MB-0001"	e.g. "Global"	Values here represent at what volumes the discount should start to be applied, e.g. • #01 = 5 • #02 = 10 • #03 = 15 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. If one of these values is missing, its volume-discount pair will be ignored.	E.g. 15% These discounts apply to ranges of volumes defined in the Volume columns. If one of these values is missing, its volume-discount pair will be ignored.

Example:



Price Parameter Values : VolumeBreakdownExceptions [1]

SKU	Dependency Level Name	Volume 1	Discount 1	Volume 2	Discount 2	Volume 3	Discount 3
MB-0007	Global	5	10.00 %	15	20.00 %	25	30.00 %

WarningConfig PP 1.7.1

For additional details, see also Warning Handling 1.7.1.

Column name	Key1	Key2	Message	Solution	Type	Alert	Matrix	Warning
Values	{error code}	• * • {DependencyLevelName}	Any text	Any text	Any text	• Empty • "Message" • "Yellow" • "Red" • "Critical"	• Yes • No	• Yes • No
Description	Code of the warning, e.g. "NO_DIMENSIONS". See their full list.	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 Result Matrix.	Defines if the warning will be registered in the default Warnings column.

Example:

Price Parameter Values : WarningConfig [58]

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

Price Setting Advanced Technical Information 1.7.1

This information is meant for Configuration Engineers if they need to modify the package.

- Standard Configuration 1.7.1
- Price Setting Manual Deployment 1.7.1
- Caching Lookup Results 1.7.1
- Batching 1.7.1
- Hot Swapping Capability 1.7.1
- Elements Documentation 1.7.1
- PriceListManagement Library 1.7.1
- Error Handling 1.7.1

Standard Configuration 1.7.1

The 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
  - DiscountGroups – Used for net price calculations
  - 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
        - AvgCompetitionAdditionalConfig
        - CostPlusAdditionalConfig
        - MaxCompetitionAdditionalConfig
        - MinCompetitionAdditionalConfigMin
        - PriceIncreaseAdditionalConfig
        - 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

Price Setting Manual Deployment 1.7.1

These steps are being replaced by deployment through PlatformManager. Until this transition is completed, you can can refer to this section for information on steps which are not yet covered in PlatformManager.

- Price Setting Repositories 1.7.1
- Create Dynamic Price Parameters 1.7.1
- Fill in Dynamically Created PPs 1.7.1
- Create Price Grid / Price List 1.7.1

Price Setting Repositories 1.7.1

First, you need to deploy these repositories:

pricefx-logic						
Calculation	Groovy libraries	SharedLib	Pricefx library with util functions.			

pricelist-mgmt-library						
Calculation	Groovy libraries	PriceListManagement	Library for performing the calculation with a specific strategy (e.g. Cost+), managing sales data lookups and other utility functionalities			
Price Parameters (with	SampleLookupEn			Sample lookup configuration		

corresponding preferences and data)	gineConfig				used by LookupEngine

pricelist-mgmt-template								
Calculation	Calculation Logics	IndependentPriceListLogic	Logic for calculating the independent level price.					
		DependentPriceListLogic	Logic for calculating the dependent level price.					
		PriceInsightsDashboardLogic	Logic used by PriceInsightsDashboard.					
	Groovy libraries	PriceBuilderCommonElementUtils	Library for common element calculations between Independent and Dependent logics.					
	Calculation Flows	CF_BuildPricingTables	Script which dynamically creates necessary Price Parameters and adds metadata to them.					
Price Parameters (with corresponding preferences and data)	AnchorAdditionalConfig			Sample table showing how to configure the AnchorEngine.				
				Sample table with anchor data for AnchorEngine. Configured in AnchorAdditionalConfig PP.				

	Anchor Data				
	Competition Additional Config			Sample table showing how to configure the CompetitionEngine. Every strategy can have attached a different copy of this parameter.	
	DependencyConfiguration			Empty table, later filled with information about dependencies. Each dependency will need to have its own Price List / Price Grid created. Entries defined in this table will show up in the configurator during PL /LPG creation. By default, a country dependency is deployed and it contains these mandatory fields: Dependency Level Name, Depends On, Source Table, Source ID, Dimension, Currency, ISO Code and SalesOrg. For SAP Customers, SalesOrg is added as a default field. Other fields can be added manually in Preference#XX columns if they are needed to connect the dependency information with other data (Cost, Transaction, etc.).	
	DependencyMapping Config			Configuration for dependency mapping mechanism used in the accelerator.	
	ExchangeRates			Table with exchange rates between currencies that will be used when recalculating prices.	
	PriceSettingConfig			Table with the package configuration, used for defining data sources, projection strategies etc. By default it is filled with the standard configuration.	
	PriceSettingDimensions			List of product parameters which is used to determine the hierarchy for dimension parameters. Currently, only products are handled. This is where the information about Product Segmentation is stored.	
	PriceSettingLevel			"Gross" - "Net / Gross" configuration (price list per dependency level).	
	Pricing Exceptions			Sample table which allows to set up price and strategy exceptions for all calculations.	
	Volum			Sample table which allows to set up volume/discount pair exceptions by SKU for volume breakdown.	

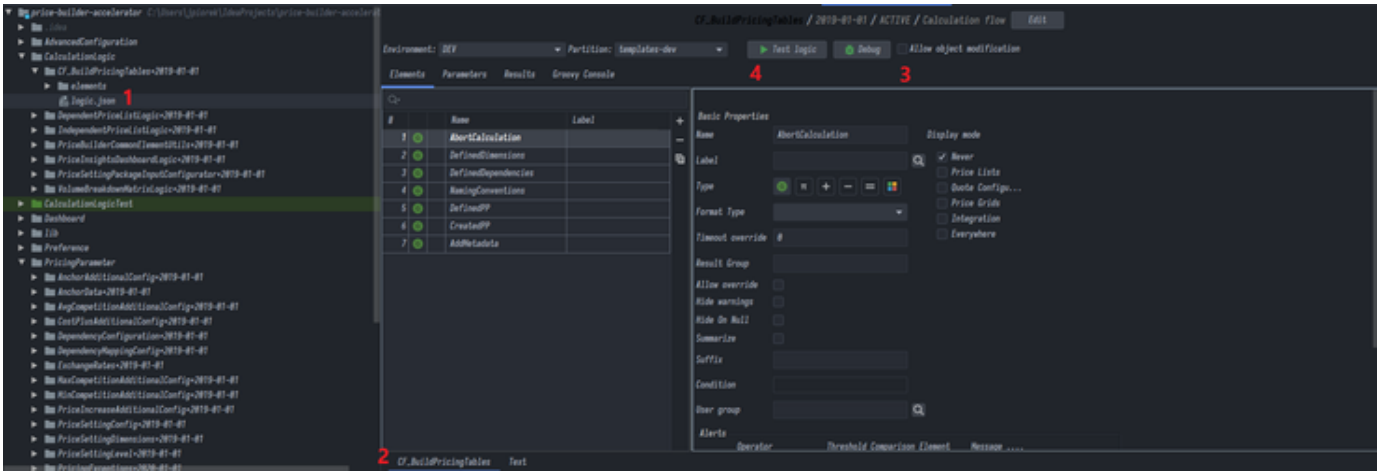
	eBreakdownExceptions				
	RoundingRulesConfig			Sample table which allows to set up the price rounding features used in the Accelerator. Empty by default.	
	StrategyDefinition			Table with the definitions of all strategies which calculations can use.	
	StrategyConditions			Table with the definitions of all conditions under which specific price strategies should be omitted.	
	WarningConfig			Table with definitions how to handle issues which can occur in the code.	
	StockData			Empty PX with all necessary fields defined so that it can be used for further configuration.	
Product Extensions (with corresponding preferences)	CostsData			Empty PX with all necessary fields defined so that it can be used for further configuration.	Optional – an existing table can be used.
	DiscountGroups				
	ListPrices				
	PromotionPrices			Empty table with a source data structure for strategy calculations.	
	Recommende			Empty table with a source data structure for strategy calculations.	

	dRetail Prices				
Dashboards	PricelnsightsDashboard			Logic behind Price Insights dashboard, for details see Price Insights Dashboard (file:///C:/wiki/spaces/ACCDEV/pages/1772716478/Price+Insights+Dashboard).	

Create Dynamic Price Parameters 1.7.1

Some Price Parameters are dependent on the previous configuration, so they are not deployed. Instead, they are created by the Calculation Flow Logic called "CF_BuildPricingTables".

The calculation flow logic cannot be run manually (like the CFS logic), so technically, there are two options: create a Calculation Flow or do it manually from the IntelliJ plugin.



1. Open the logic.json file of CF_BuildPricingTables in IntelliJ IDEA.
2. At the bottom of the window, choose "CF_BuildPricingTables".
3. Enable "Allow object modification".
4. Click "Test logic".

Every price parameter that will be created will have keys corresponding to the values defined in the "PriceSettingDimensions" PP. The created PPs and their values will be:

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

    - Price Strategy #5
    - Prioritize Independent Level Price
- MinMargin
    - Min Margin %
- {nameOfDependency}DependencyLevelAdjustment – one for every dependency defined in "DependencyConfiguration"
    - Adjustment %
- { nameOfDependency }DiscountLevel – one for every dependency defined in "DependencyConfiguration"
    - Discount Level %
- { nameOfDependency }VolumeBreakdown – one for every dependency defined in "DependencyConfiguration"
    - Volume #01
    - Discount #01
    - ...
    - Volume #15
    - Discount #15
- CostPlus
    - Plus %
- PriceIncrease
    - Increase %
- AdjustedPriceCorridor
    - Minimum Absolute
    - Minimum Corridor
    - Maximum Corridor
    - Maximum Absolute
- ListPriceCorridor
    - Minimum Absolute
    - Minimum Corridor
    - Maximum Corridor
    - Maximum Absolute
- RelevantCompetitionData
    - Competitor #01-#29

⚠ Due to technical reasons, these PPs:

- Can only be created in the global directory. They can be, however, easily moved manually.
- Do not have preferences after deployment. They can be, however, set manually.

Fill in Dynamically Created PPs 1.7.1

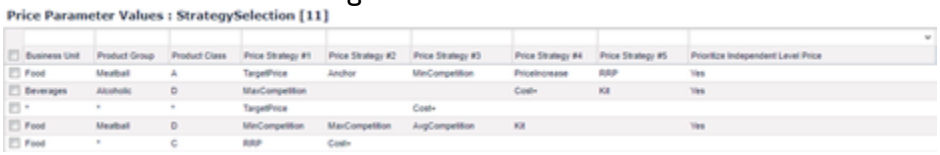
Dynamically created Price Parameters are standard tables used only at the calculation level. Instead of a standard lookup, the hierarchical lookup is performed, which allows generalizing some groups of products.

Keys

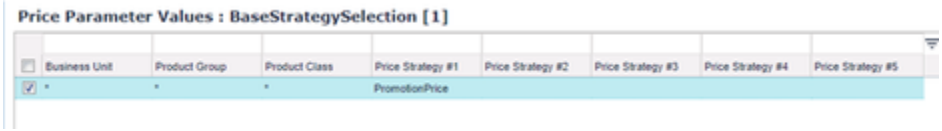
Keys are the parameters for hierarchical lookup. The asterisk * will match everything. If there are 5 dimensions defined, the lookup will be performed with all 5 keys. If no row is found, the last key will be omitted and the lookup will be performed again. That is why keeping one row with only "*" in the key is a good way to provide fallback on all non-standard products.

Values

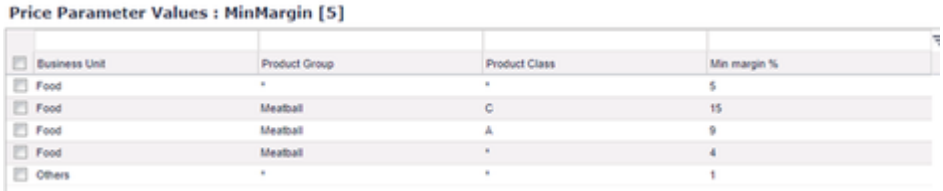
- **StrategySelection –** List of price strategies calculated for the given product. "Price Strategy #1" will be treated as the default on the independent level price list. On a dependent level price list, the independent level price will always be prioritized, unless there is "No" in "Prioritize Independent Level Price". Default strategies can be overwritten at PLI/PGI level.



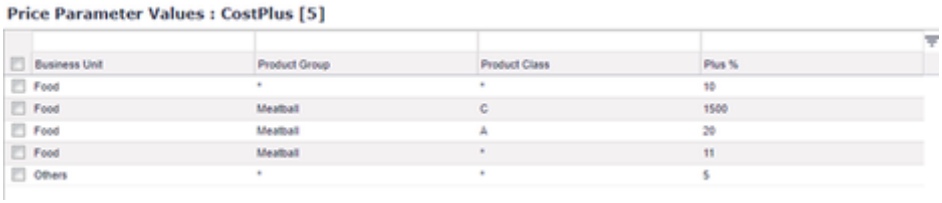
- **BaseStrategySelection** – List of optional strategies calculated for the given product. They will always appear before strategies in StrategySelection. The difference is that they will be hidden if the engine returns an error.



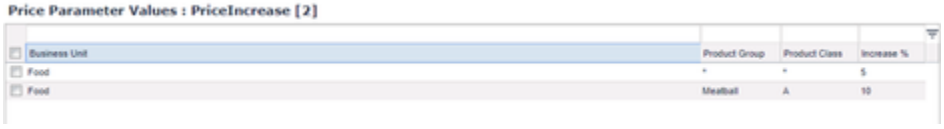
- **MinMargin** – Defines when an action should be performed because of too low margin. Currently api. redAlert() is called.



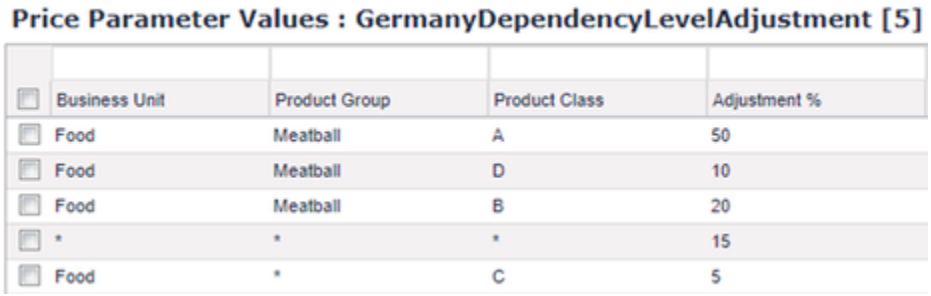
- **CostPlus** – Percentage amount of an increased value in the "Cost+" strategy.



- **PriceIncrease** – Percentage amount of an increased value in the "PriceIncrease" strategy.



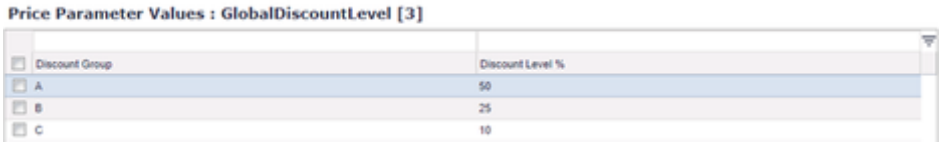
- **{nameOfDependency}DependencyLevelAdjustment** – Defines how the independent level price should be affected in a specific dependency for a specific product.



Price Parameter Values : GermanyDependencyLevelAdjustment [5]

Business Unit	Product Group	Product Class	Adjustment %
Food	Meatball	A	50
Food	Meatball	D	10
Food	Meatball	B	20
*	*	*	15
Food	*	C	5

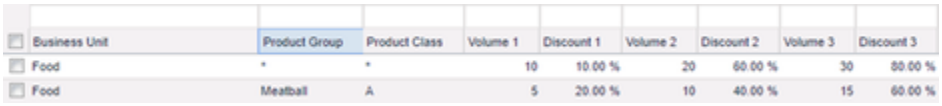
- **{nameOfDependency}DiscountLevel** – Defines the discount % for each discount group of products for a specific dependency level name.
  Note: It is impossible to use the * fallback in these parameters as in other parts of the package. We consider the discount structure to be a critical and sensitive part of the master data, so we need to be exact about what we put there.



Price Parameter Values : GlobalDiscountLevel [3]

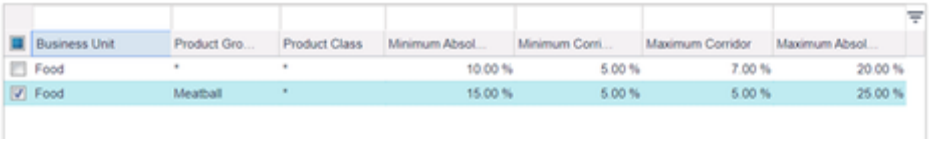
Discount Group	Discount Level %
A	50
B	25
C	10

- **{nameOfDependency}VolumeBreakdown** – Defines the volume discount ranges for various product dimensions. Used in the Volume Breakdown feature.



Business Unit	Product Group	Product Class	Volume 1	Discount 1	Volume 2	Discount 2	Volume 3	Discount 3
Food	*	*	10	10.00 %	20	60.00 %	30	80.00 %
Food	Meatball	A	5	20.00 %	10	40.00 %	15	60.00 %

- **AdjustedPriceCorridor/ListPriceCorridor** – Defines the percentage values of price corridors. Rows are defined based on the lookup keys. Initially, these parameters are deployed due to the bootstrapping mechanism. Before filling the values, we need to change the format type of columns which will contain values of the corridors Minimum Absolute, Minimum Corridor, Maximum Corridor, Maximum Absolute to Real Percentage.



Business Unit	Product Gro...	Product Class	Minimum Absol...	Minimum Corri...	Maximum Corridor	Maximum Absol...
Food	*	*	10.00 %	5.00 %	7.00 %	20.00 %
Food	Meatball	*	15.00 %	5.00 %	5.00 %	25.00 %

- **CostSelection** – Defines in which order the advanced cost will be calculated.

Price Parameter Values : CostSelection [1]

Business Unit	Product Group	Product Class	Cost Type #1	Cost Type #2	Cost Type #3	Cost Type #4	Cost Type #5
*	*	*	Cost3	Cost2			

Create Price Grid / Price List 1.7.1

The last configuration task is to create Price Lists or Price Grids for an independent calculation and optionally one for every dependent region.

For a independent/standalone Price List, the selected logic should be "IndependentPriceListLogic". For other, "DependentPriceListLogic" should be used.

To use the Volume Breakdown 1.7.1 functionality, you need to create a Price List or Price Grid of the MATRIX type, then in the Matrix logic select VolumeBreakdownMatrixLogic and in the Matrix logic element select Volumes.

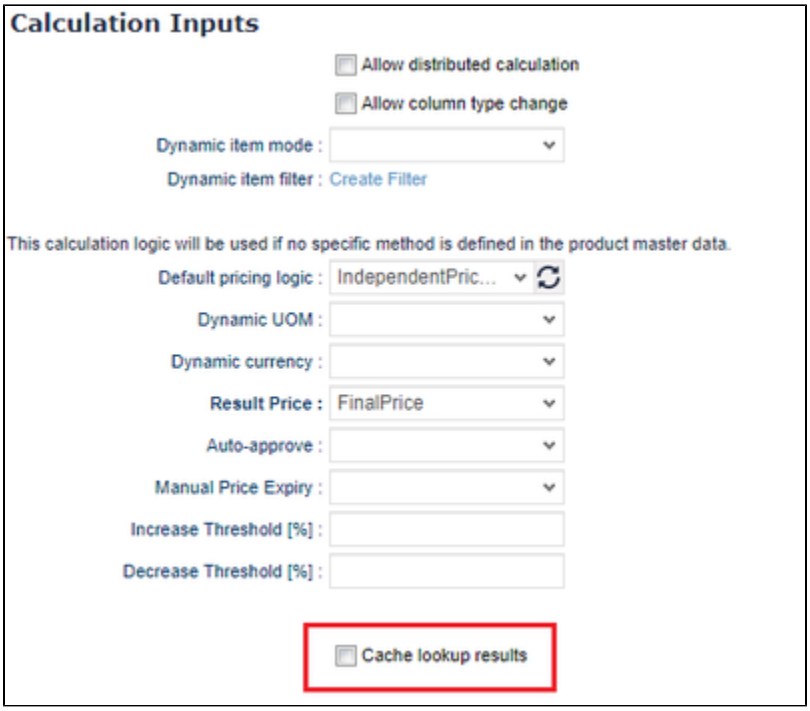
In the Price Lists / Price Grids, hide the Product Currency column (it gets the information from Product Master and it is not needed here it; currency is taken from the Currency column). To hide a column, use Preferences. You can also set default preferences for a price list in Pricefx Configuration.

Caching Lookup Results 1.7.1

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 1.7.1.

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:



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 1.7.1

Our own BatchManager, built on top of SharedLib's batchUtil is used for batching. Batching is performed for:

- Transaction data
- Competition data
- Actual price data

Hot Swapping Capability 1.7.1

Hot swapping means changing configuration without damaging the feature itself. With that in mind, we can split configuration into the following parts:

- General Advice
- Dynamically Created PPs
- PriceSettingLevel
- PriceSettingConfig
- Other PPs

General Advice

- 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 the configuration column (it might be especially tempting in Dependency Configuration). Column names of sample data might be freely changed, assuming that the configuration will be properly adjusted.

Dynamically Created PPs

Dynamically created Price Parameters can be changed without redeployment. As described in Create Dynamic Price Parameters 1.7.1, these PPs act as standard Pricefx Price Parameters:

- BaseStrategySelection
- StrategySelection
- MinMargin
- CostPlus
- {nameOfDependency}DependencyLevelAdjustment - one for every dependency defined in DependencyConfiguration
- {nameOfDependency}DiscountLevel - one for every dependency defined in DependencyConfiguratior
- PriceIncrease
- AdjustedPriceCorridor
- ListPriceCorridor

Values can be added/removed on demand. We recommend to have one value with an asterisk "*" on every key as a way to provide a fallback if the lookup does not match any specific row.

PriceSettingLevel

This Price Parameter may be changed, but creating a new PL/PG is required after such an action.

PriceSettingConfig

PriceSettingConfig can be changed with extreme caution. Every part of PriceSettingConfig can be changed during the system run but the changes affect much more than the first group. That is why you should be careful here and customers are not even recommended to do this on their own.

Discount Group Config / Transaction Config / Projection Config

Changes in these configs affect sources from which the user data is read. In case of the Projection config, it also impacts how the projection is calculated. Because of the common character of this configuration, all calculations are affected.

Make sure that:

1. There are no typos.
2. Every table has its real name (instead of a label).
3. The source table is properly prepared (with proper labels and data).

⚠ Any mistake here can break Last Year, YTD and Forecast values in all price lists.

Cost Config

Changes in this config affect calculations for every strategy that uses "Cost" as an input value. This config is applicable to all price lists.

Actual Price Lookup

Changes in this setting affect calculation of the PriceIncrease strategy. This setting is applicable to all price lists.

Independent Manual Override / Dependent Manual Override / Price Exception / Strategy Exception

These rows define what kind of overrides are available. Changes in these values make some fields hidden /shown and because of the way Pricefx treats such changes, all 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.

Other PPs

Other Price Parameters cannot be changed.

The following Price Parameters should never be changed without redeploying a feature from scratch:

- PriceSettingDimensions
- DependencyConfiguration

Elements Documentation 1.7.1

- Calculation Information Elements 1.7.1
- Final Price Elements 1.7.1
- Manual Override Elements 1.7.1
- Price Calculations Elements 1.7.1
- Price Checks Elements 1.7.1
- Price Insights Dashboard Logic Element 1.7.1
- Product Information and Calculation Types Elements 1.7.1
- Sales Data Elements 1.7.1
- Technical Elements 1.7.1

Calculation Information Elements 1.7.1

Element Name	Independent Price List	Dependent Price List	Visible for User	Return Type	Description
Warnings	Yes	Yes	No	Result Matrix	User friendly table showing all warnings which occurred during the calculation.

PriceBuilderCommonElementUtils

Element name	Return Type	Description
BatchManager	Utility element	The interface used for unified batching of data loads.
ConstConfigs	Utility element	Contains constant configs that are used in the configuration element.
ConfigUtils	Utility element	Contains functions that are used in the config creation process.
DependencyMapping Utils	Utility element	Contains functions correlated with the dependency mapping mechanism.
TransactionUtils	Utility element	Contains functions correlated with the transaction data subject.
TransactionDataUtil	Utility element	Contains functions used for transaction data loading and preprocessing.
ConfigManager	Utility element	The interface between configurations which is passed to the calculation logic elements.
ExceptionsManager	Utility element	Contains functions correlated with exceptions.
FormatingUtil	Utility element	Contains functions correlated with formatting prices.
Library	Utility element	Contains commons functions used in both independent and dependent logics.
ListPriceLookupUtils	Utility element	Contains functions correlated with the list price lookup mechanism.
StockUtils	Utility element	Contains functions for handling product stock data.
RoundingUtils	Utility element	Contains functions for rounding data.
RoundingRulesConfig Manager	Utility element	The interface used for getting related information for rounding.

Final Price Elements 1.7.1

These elements are used to determine which price is the Final Price.

Element Name	Independent Price List	Dependent Price List	Visible for User	Return Type	Description
UsedListPrice	Yes	Yes	No	BigDecimal	Value selected in PriceSelector or inserted in ManualPrice.
FinalListPrice	Yes	Yes	Gross / Net only	BigDecimal	Value copied from UsedListPrice.
Discount	Yes	Yes	Gross / Net only	BigDecimal	Percentage discount applied when calculating the transition from a gross to net price.
VolumeDiscount	Yes	Yes	Yes	BigDecimal	Percentage discount applied after all price calculations and exceptions, used in the VolumeBreakdown feature.
NetPrice	Yes	Yes	Gross / Net only	BigDecimal	UsedListPrice with a discount applied.
FinalPrice	Yes	Yes	Yes	BigDecimal	UsedListPrice in case of a Gross calculation; NetPrice in case of a Gross/Net calculation.
PriceDecision	Yes	Yes	Yes	String	Reason for a given price. ManualPriceReason if the price was overridden, pricing strategy from PriceSelector otherwise. If there was an ExceptionsTable exception, the value is "ExceptionTable - <exception type>".
Margin	Yes	Yes	Yes	BigDecimal	Margin calculated based on FinalPrice and product cost.
MinimumMargin	Yes	Yes	Yes	BigDecimal	Minimum margin retrieved from the RawMinimumMargin element with proper warning handling.

Manual Override Elements 1.7.1

These elements allow pricing managers to override the calculated prices.

Element Name	Independent Price List	Dependent Price List	Visible for User	Return Type	Description
AllowStrategyOverride	Yes	Yes	No	Boolean	Defines whether the PriceSelector element should be visible.
AllowPriceOverride	Yes	Yes	No	Boolean	Defines whether the ManualPrice and ManualPriceReason elements should be visible.
ExceptionsManager	Yes	No	No	Map<String, Closure>	Interface used for managing exceptions, manual overrides and making them allowed. Exceptions/Allowance logic is implemented here.
PriceManager	Yes	Yes	No	Map<String, Closure>	Interface used for centralized price calculations.
PriceSelector	Yes	Yes	Conditional	AttributeResult	Dropdown menu with calculated prices and their strategies. Users can override a default strategy here.
ManualPrice	Yes	Yes	Conditional	String	Empty field acting as an input field for the user. Fill it with an overridden price if necessary. Visibility depends on the configuration in ExceptionsManager. This value pops up automatically if there is an ExceptionsTable price exception.
ManualPriceReason	Yes	Yes	Conditional	String	Empty field acting as an input field for the user. Fill it with a reason why a price has been overridden. Visibility depends on the configuration in ExceptionsManager. If there is an ExceptionsTable price exception, the value is "Price Exception".
Exceptions	Yes	Yes	Yes	String	Message with exceptions (if any). The value will be: • Price Exception: <value> • Strategy Exception: <value> • Price Exception: (+1) <value> – if both exceptions are present
OverrideRemover	Yes	Yes	No	N/A	Used for clearing manual/table overrides/exceptions when the default strategy does not allow it.

Please 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 Calculations Elements 1.7.1

These elements perform calculations based on configured strategies.

Element Name	Independent Price List	Dependent Price List	Visible for User	Return Type	Description
IndependentLevelAdjustedPrice	No	Yes	No		FinalPrice from the independent level price list, adjusted with the dependent level adjustment.
AdditionalCalculatorParameters	Yes	Yes	No	Map< String, Map>	Place for CS to inject their own variables into the calculator.
CalculatedPrices	Yes	Yes	No	List	Calculated prices for every defined strategy.
StrategyCondition	Yes	Yes	No		Applies defined pricing rules (defined in Conditions 1.7.1) to the current list of prices.
IndependentLevelAdjustedPrices	No	Yes	No		Prices from the independent level price list, adjusted with the dependent level adjustment.
PricesSummary	No	Yes	No	List	Generates data for a popup and priceSelector, also collects prices from the independent level price list and applies a dependent level adjustment to it.
PopUpData	Yes	Yes	No	List	Calculates marginValues for a popup data input.
Prices	Yes	Yes	Yes	Matrix Result	PopUpData formatted into a user friendly MatrixResult.

Price Checks Elements 1.7.1

The following configurable checks will be performed:

Element Name	Independent Price List	Dependent Price List	Visible for User	Return Type	Description
MinimumMargin	Yes	Yes	Yes	BigDecimal	Minimum margin for a specific product checked against Margin.
AdjustedPriceCorridor	No	Yes	Yes	Percent	Deviation between the independent level adjusted price and FinalList price.
ListPriceCorrior	No	Yes	Yes	Percent	Deviation between the independent level price and FinalList price.

Price Insights Dashboard Logic Element 1.7.1

Element Name	Return Type	Description
InputSKU	String	Input element for the user to pass a SKU into the logic.
ShouldFilterEmptyValues	Boolean	Input element for filtering empty results.
Utils		Container for commonly used methods.
PriceLists	List<PL>	Returns only approved price lists which use the defined calculation logics.
PriceListsItems	Map	Extracts data from a price list for a defined SKU.
PriceGrids	List<PG>	Returns price grids which use the defined calculation logics.
PriceGridsItems	Map	Extracts data from a price grid for a defined SKU.
TransactionData	Map	Generates AVG, MIN, MAX invoice prices based on the last year invoices.
Chart builder		Builds a chart with the defined data and config.

Product Information and Calculation Types Elements 1.7.1

These elements are input parameters used by the calculations.

Element Name	Independent Price List	Dependent Price List	Visible for User	Return Type	Description
InputConfigurator	Yes	Yes	Yes	String	Input configurator entry element that allows the user to select dependency options for a given calculation. • If the "Independent mode" is used, the user will be presented with the dropdown to select the level. • If the "Dependent mode" is used, the user will be presented first with the dropdown to select the independent level on which the calculation will be based and then the dependent level. The list of available options is defined in DependencyConfiguration.
ChangesFromMonitor	Yes	No	Yes	String	If isPLCMTarget is true, it contains ResultMatrix describing to the user why this element was added to the Price Grid. Currently supported only for Independent Level LPGs.
DependencyLevelName	Yes	Yes	No	String	Dependency level for which the current calculation runs.
DependsOn	No	Yes	No		Independent level on which this dependent calculation depends on.

				String	
DependencyLevelLookup	Yes	Yes	No	LTV	Row from the "DependencyConfiguration" PP.
ActualPriceLookup	Yes	Yes	Yes	BigDecimal	Lookup for the current product price which is fetched based on the configuration.
BaseCurrency	Yes	Yes	No	String	Currency of the price list/grid.
IndependentLevelCurrency	No	Yes	No	String	Currency of the independent price list. The data is loaded from the "DependencyConfiguration" PP.
NetPriceLevel	Yes	Yes	Yes	Int (as bool)	Indicates if the net price will be calculated.
IndependentLevelCalculatedItem	No	Yes	No	PGI or PLI	Calculation result for this product on the independent level.
ExchangeRate	No	Yes	No	BigDecimal	Exchange rate between the independent level and current dependent level currencies.
IndependentLevelDecision	No	Yes	Yes	String	Decision extracted from IndependentLevelCalculatedItem. Can be a strategy name, an exception table type message or a manually entered reason.
IndependentLevelPrice	No	Yes	Yes	BigDecimal	Price from the independent level calculation result extracted from IndependentLevelCalculatedItem, after applying exchange rates.
DependentLevelAdjustment	No	Yes	Yes	BigDecimal	Percentage by which independent level prices will be multiplied to obtain dependent level prices.
DimensionalLookupKey	Yes	Yes	No		Stores values of calculated product's dimensions.
Cost	Yes	Yes	Yes	BigDecimal	Cost of the product read from PX configured in "PriceSettingConfig" PP. If not found, null is returned.
RawCompetitionData	Yes	Yes	No	Map	Data about competitor pricing, read from the PCOMP table.
RawRelevantCompetitionData	Yes	Yes	No	List	List containing relevant competitors data.
CompetitionData	Yes	Yes	Yes		Formatted data about competitor pricing, read from the PCOMP table.

				Result Matrix	
RelevantCompetitionData	Yes	Yes	Yes	Result Matrix	Formatted relevant competitors data.
RawMinimumMargin	Yes	Yes	No	Big Decimal	Data used for Minimum Margin calculation.
MinimumMarginPrice	Yes	Yes	Yes	Big Decimal	Price calculated from Minimum Margin, this is the minimum affordable price.
Volume	Yes	Yes	No	Integer	Volume read from the VolumeBreakdown PP table, used in the VolumeBreakdown feature.
LookupKeys	Yes	Yes	No	Map	Keys in the format Map<String, List<String>> where the key is a dimension and the List is a sorted list of keys (dimensions) for hierarchical lookups. Currently, only the Product dimension is supported.
PriceStrategies	Yes	Yes	No	List of Strings	Ordered list of possible pricing strategies for the current product, read from the StrategySelection PP.
BaseStrategies	Yes	Yes	No	List of Strings	Ordered list of base strategies that precede normal price strategies. Used e.g. to make promotions override the standard strategy config.
DiscountGroup	Yes	Yes	Gross / Net only	String	Discount group for a specific product for a specific dependency level.
StrategySelectionEntry	No	Yes	No	List of Strings	Lookup result from the StrategySelection PP. It exists only in the dependency price list, since only there more than one element can utilize this result.
Stock	Yes	Yes	Yes	Big Decimal	The stock of the product read from PX/PP configured in the PriceSettingConfig PP. If not found, null is returned.
StockCoverDays	Yes	Yes	Yes	Big Decimal	The calculated stock cover days of the product.

Sales Data Elements 1.7.1

The calculation shows not only a price, but also sales data for the last year and forecast data for the next year.

Element Name	Independent Price List	Dependent Price List	Visible for User	Return Type	Description
TransactionData	Yes	Yes	No	null	Cached map with transaction data for every product. Results are split into the last year and year to date.
Forecast Data	Yes	Yes	No	null	Cached map with forecast data for every product. It is calculated based on PP configuration.
SalesVolumeYTD	Yes	Yes	Yes	Long	Volume from this year.
Turnover YTD	Yes	Yes	Yes	BigDecimal	Turnover from this year.
LastYear SalesVolume	Yes	Yes	Yes	Long	Volume from last year.
LastYear Turnover	Yes	Yes	Yes	BigDecimal	Turnover from last year.
SalesVolumeForecast	Yes	Yes	Yes	Long	Volume forecast for the upcoming period with values taken from a specific strategy set in the PriceSettingConfig PP.
Turnover Forecast	Yes	Yes	Yes	BigDecimal	Turnover forecast for the upcoming period with values taken from a specific strategy set in the PriceSettingConfig PP.

Technical Elements 1.7.1

These elements are used just for the logic. Business users can skip this part.

Element Name	Independent Price List	Dependent Price List	Visible for User	Return Type	Description
RetainGlobal	Yes	Yes	No	N/A	Standard element to utilize the cache.
CacheLookupKeys	Yes	Yes	No	Boolean	Keeps the value for the lookup caching configuration.
Configuration	Yes	Yes	No	Map	Cached interface to all core configurations used in Price Setting Package calculations.
Library	Yes	No	No	N/A	Place for utility functions used only in a given logic.
IsPLCMTarget	Yes	No	No	Boolean	Checks if the current Live Price Grid is defined as a target in the Price Flexibility package.
AbortCalculation	Yes	Yes	No	N/A	Aborts calculation in the syntax check mode.
BaseWarns	Yes	Yes	No	N/A	Performs only basic checks if dependent logics are deployed.
ModuleManager	Yes	Yes	No	Map	Cached interface to get single module manager configs.

TransactionModuleIsOn	Yes	Yes	No	Boolean	Keeps information if the transaction module is installed.
NetPriceModuleIsOn	Yes	Yes	No	Boolean	Keeps information if the net price module is installed.
ExceptionsModuleIsOn	Yes	Yes	No	Boolean	Keeps information if the exceptions module is installed.
PriceChecksModuleIsOn	Yes	Yes	No	Boolean	Keeps information if the price checks module is installed.
FlexibilityModuleIsOn	Yes	No	No	Boolean	Keeps information if the Price Flexibility module is installed.
StrategyConditionsModuleIsOn	Yes	Yes	No	Boolean	Keeps information if the strategy conditions module is installed.
ProductCompetitionModuleIsOn	Yes	Yes	No	Boolean	Keeps information if the product competition module is installed.
RoundingRulesModuleIsOn	Yes	Yes	No	Boolean	Keeps information if the rounding rules module is installed.
AdvancedCostModuleIsOn	Yes	Yes	No	Boolean	Keeps information if the advanced cost module is installed.
TransactionConfigManager	Yes	Yes	No	Map	Cached interface to all transaction related configurations used in Price Setting Package calculations.
NetPriceConfigManager	Yes	Yes	No	Map	Cached interface to all net price related configurations used in Price Setting Package calculations.
ExceptionsConfigManager	Yes	Yes	No	Map	Cached interface to all exceptions related configurations used in Price Setting Package calculations.
PriceChecksConfigManager	Yes	Yes	No	Map	Cached interface to all price checks related configurations used in Price Setting Package calculations.
FlexibilityConfigManager	Yes	No	No	Map	Cached interface to all Price Flexibility related configurations used in Price Setting Package calculations.
StrategyConditionsConfigManager	Yes	Yes	No	Map	Cached interface to all strategy conditions related configurations used in Price Setting Package calculations.
ProductCompetitionConfigManager	Yes	Yes	No	Map	Cached interface to all product competition related configurations used in Price Setting Package calculations.
RoundingRulesConfigManager	Yes	Yes	No	Map	Cached interface to all rounding rules related configurations used in Price Setting Package calculations.
AdvancedCostConfigManager	Yes	Yes	No	Map	

					Cached interface to all advanced cost related configurations used in Price Setting Package calculations.
WarningManager	Yes	Yes	No	Map<String, Closure>	Creates an object to handle warnings.
Batch	Yes	Yes	No	Map<String, Closure>	Creates an object to handle batching.

PriceListManagement Library 1.7.1

PriceListManagement is a library which can perform various tasks. This library is independent from other calculation logics.

- General Utility Elements 1.7.1
- Calculation Oriented Elements 1.7.1

⚠ Be careful when changing PriceListManagement. There are various connections among the elements and their order does not matter.

General Utility Elements 1.7.1

Used by calculation engines and both calculation logics. Can be used separately by other features as well.

Element Name	Description
DateUtils	Utility functions to work with dates.
ProductExtensionUtil	Utility functions to read data from PX tables. Some functions use labels to filter dependencies, so library specific maps are needed.
PriceParameterUtil	Utility function to find and work with values in the hierarchical structure of dimensions.
DataStructureUtils	Utilities used for data transformation.
LookupUtils	Utilities used by some PGI/PLI calculations.
ErrorMapping	Common error codes and messages templates used by different engines.

Calculation Oriented Elements 1.7.1

Price calculation engines and necessary utilities. Can be used separately by other features as well.

Element Name	Description
PriceCalculator	Calculation dispatcher that runs calculation engines based on the configuration provided in the StrategyDefinition PP.
	Engine used for Cost+ and Price Increase strategies.

AdjustmentEngine	
CompetitionEngine	Calculation for Competition Based Strategy.
LookupEngine	Engine for existing price lookups, e.g. PromotionPrices or RRP.
NetEngine	Used for calculating a price that will be equal to the target price after applying the discounts.
KitEngine	Calculates kit prices based on the Bill of Material data.
AnchorEngine	Calculates prices based on another product's price (anchor).
CalculationUtils	Util function for various strategies.

Error Handling 1.7.1

Error handling is structured. All errors should be expected and they are all defined in WarningConfig with instructions what to do when such an error happens.

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.

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

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".

Price Setting Package Administration Procedures 1.7.1

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

- Renaming Dependency Level Name 1.7.1

Renaming Dependency Level Name 1.7.1

⚠ 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** a fter the renaming.

Release Notes PSP 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