Agreements Accelerator

Version 0.1.0

January 2024

Accelerate Agreements Package

The Agreements Accelerator provides a solution for faster implementation of formula-based long-term customer agreements. Its purpose is to facilitate the creation and renewal of these agreements through the utilization of both a formula library and a formula-based Agreement Type. In combination, they allow you to generate agreement pricing utilizing different scenarios where you can define different formulas and parameter combinations across your product portfolio.

The package includes:

- Formula library
- Formula-based Agreement Types with predefined Header, Scenario, and Line Item Inputs
- Approval workflows
- Output template for generated prices
- Recalculable agreements

Key Capabilities

- Create and manage a set of formulas to enable dynamic pricing of commodities and specialties
- Establish agreements with multi-annual terms with the option to set the frequency of price recalculation
- Define calculation rules to adjust prices based on specific situations:
  - Bounding rules
  - Pricing exceptions
- Push resulting agreement price records for use within a given IT landscape (ERP, CRM, etc.)

In this section:

Business User Reference (Agreements)

Agreements Accelerator provides support for formula based agreement pricing and allows you to periodically recalculate agreements.

Using this accelerator you get a head start in creating agreements and agreement renewals with regular recalculation of prices.

- Define Formula
- Apply Formula to Agreement Line Item
- Calculate or Recalculate

Define Formula

To make the accelerator work, we need:

- **Formula type** – Stands for a calculation engine; it defines a way of calculating. Currently, the only provided type is *Cost plus*.
- **Formula definition** – Stands for a specific setup of the engine.

One formula type can be used by multiple formula definitions and formula definitions provide parameters to the engine, such as lookup parameters, weights, etc.

Formula Type

Formula Types are defined in the Company Parameter table called AGR_FormulaTypeDefinitions. There are 4 columns:

- **Formula Type Name** – Specifies the name of the formula.
- **Calculation Engine Path** – Specifies how the engine is calculated. Provided in the format *libs.<name of lib>.<name of element>.<name of method>,* for example: libs.AGR_FormulaEnginesLib. AGR_Formula_CostPlus.calculate
- **Engine Calculation Parameters** – Parameters for the Calculation Engine, some of them are predefined, most of them are input names from Input Manager.
- **Input Generation Table Name** – Name of the Company Parameter table that contains definitions of inputs to be shown on the Formula Definition (as well as broadcasted to the Agreement). The table has to follow the Input Manager input definition table setup.

Formula Definition

1. Go to **Agreements & Promotions** > **Formula Definition**.
2. Click **Create New Formula Definition**.
3. Select what **Formula Type** (calculation engine) the definition should be based on.
   Currently, the only option available is *AGR_Formula_CostPlus*, so all of these options relate to it.
   - Cost Plus general fields:
     - Select the type of the structure where your **data** is located. Currently, the only supported data source for costs is *PX (Product Extensions)*.
     - Select the name of the **table** where you have your cost data. Currently, it has to be PX table.
     - Select the column where the **Cost** is stored.
     - Select the column where the **Currency** is stored.
     - Select the column where the **UoM** is stored.
     - Select the column where **Valid From** is stored.
     - Select the column where **Valid To** is stored.
   - Cost Plus calculation specific fields:
     - Select **Adder Type** – either ABSOLUTE or PERCENTAGE.
     - **Base Adder** – Specifies the amount (either as an absolute value or percentage) which should be added to the price. It uses the default currency and default UOM defined at the header of an agreement.
       Note: You can override the adders on line item level in the agreement and using exceptions for specific products.
     - **Bounds** – Represent a mechanism to prevent too frequent price changes – prices get updated only if the price change is big enough. Bounds use comparison with the *previous result* which is the engine result of the previous period for the same agreement and the same SKU using the same currency. The previous result is then compared to the *calculated result* for the current period and their difference is calculated (previous minus calculated). If the

difference is greater than the defined bounds, the newly calculated price will be used. Both bounds are specified as positive numbers. See an example below.

- **Upper Bound** – The newly calculated price is used only if it *grew* more than the amount specified here.
- **Lower Bound** – The newly calculated price is used only if it *dropped* more than the amount specified here.
  Note: You can override the bounds with exceptions for specific products.

- **Rolling Period** – Allows you to calculate an average price over the defined period of time. For example, if Rolling Period is 3 months, it looks at the current date and then takes the previous 3 months *plus the current one* and uses the costs found in those 4 months for calculating the average. If Lag Period (see below) is defined too, it is applied first. That means that first the current date is moved by the defined number of months/quarters and only then the previous months/quarters are looked up.
  - **Rolling Period Type** – Either Month or Quarter.
  - **Rolling Period Amount** – Defines over how many months/quarters to calculate the average price.
- **Lag Period** – Moves your cost lookup by the defined number of months/quarters back from the current date.
  - **Lag Period Type** – Either Month or Quarter.
  - **Lag Period Amount** – Defines how many months or quarters to move back.

4. Once you finish the Formula Definition, you need to submit it for approval.

Bounds Examples

Example 1	Example 2	Example 3
Scenario: • Previous price = 12 • New calculated price = 14 • Upper Bound = 5 • Lower Bound = 2	Scenario: • Previous price = 12 • New calculated price = 19 • Upper Bound = 5 • Lower Bound = 2	Scenario: • Previous price = 12 • New calculated price = 9 • Upper Bound = 5 • Lower Bound = 2
Result: Difference between previous and calculated price = 2 It is a price increase, so we apply Upper Bound. The price difference is not bigger than the Upper Bound value and so the previous price stays in place.	Result: Difference between previous and calculated price = 7 It is a price increase, so we apply Upper Bound. The price difference is bigger than the Upper Bound value and so the new calculated price is used.	Result: Difference between previous and calculated price = 3 It is a price drop, so we apply Lower Bound. The price difference is bigger than the Lower Bound value and so the new calculated price is used.

Apply Formula to Agreement Line Item

1. Go to **Agreements & Promotions** > **Agreements & Promotions**.
2. Expand the **New Agreement & Promotion** button and select the option **Formula Based Pricing**.
3. In the header fill in the following fields:
   - **Start/End Date** – Defines validity period of the agreement. According to this period, agreements to recalculate are selected (by feeder logic in Calculations).

- **User Group (View Details / Edit)** - If the agreement's visibility and editability should be restricted to certain user groups, enter their names.
- **Customer** – Currently not used.
- **Default Currency / UOM** – Define currency and UOM for this agreement.
- **Calculation Output Types** – Allow you to specify additional parameters for the result. For example, you can have the price generated in another currency or using different bounds (explained below). For each combination, a set of price records is generated.

4. On the Items tab, there is a predefined scenario and predefined line item. (One of each must always be present in the agreement. If you delete the scenario and click Recalculate, it will be populated back.) Use this line or create your own.
   - In the table you have the following options for a scenario:
     - **Active** checkbox indicates that this scenario should be used for recalculations and for price record generation.
     - **Calculation Period** specifies how often the price records are recalculated: monthly or quarterly. The logic looks through all contracts, finds those with active scenarios and then checks their calculation periods. If there is a price record generated in the current month /quarter, then no generating takes place. But if there isn't one, then a new price record for the current month/quarter is generated.
     - Note that **Default Currency** and **Default Unit of Measure** are propagated from the header. If they are changed in the header, the change will be reflected here only after recalculation.
5. Add required line items into a scenario. For each item fill in the following fields:
   - Input parameters (panel on the right side):
     - **Product(s)** – Specifies what Product Group will provide SKUs to use for the recalculation. The number of products and the number Calculation Output Types directly translate into how many price records are generated.
     - **Formula** – Allows you to select the appropriate formula definition (has to be approved).
     - The following inputs (Adder Type, Base Adder, Exceptions) are broadcasted from the Cost Plus formula, so they will not always be here, if a different formula is used.
       - **Override Adder Type** – Allows you to enter a different Adder Type than in the formula definition.
         Note that this value can still be overridden if you define an exception for selected products (see below).
       - **Override Base Adder** – Allows you to enter a different Base Adder than in the formula definition.
         Note that this value can still be overridden if you define an exception for selected products (see below).
       - **Product Exceptions** – Allows you to specify products which should be treated differently (different adders, different bounds, different currency). The priority then is: 1. Exceptions, 2. overrides, 3. base (the one in formula definition).
6. Once the agreement is recalculated, you can go to the Calculations tab of the panel on the right. It shows **Calculation Outputs Result Matrix** which summarizes all the combinations: SKU, resulting price, currency, UOM, formula detail and message.

Calculate or Recalculate

All (re)calculations are done using the default currency and default UOM (those are defined at the header of an agreement). That implies that you will get the results only if your PX table has the cost defined for the default currency/UOM.

1. Go to **Agreements & Promotions** > **Calculations** and open the "Agreement Recalculation" Calculation object.
2. Open the calculation and go through all its tabs:
   - **Schedule** – Allows you to set up how often the recalculation should run.
   - **Agreements & Promotions** – Allows you to define a lists of agreements to recalculate. The agreement must be approved in order to be recalculated.
     Note: Selection by the feeder logic (see below) has priority over whatever is selected here.
   - **Calculation** – Allows you to select the logic and the feeder logic. The feeder logic provides a list of agreements to recalculate. These agreements must fulfill these conditions:
     - Have "Formula based pricing" as Header Type.
     - Be approved.
     - Be valid (fit in the range of from-to dates).
     - If there is a price record generated in the current month/quarter, then no generating takes place. But if there isn't one, then a new price record for the current month/quarter is generated.
     - If the agreement has a Price Record with an error, it is recalculated.
3. Once everything is set up, click the **Save** button.
4. Let the calculation run according to the schedule.
   Or, if you want to trigger the calculation manually, click **Run Calculation**.
5. Only after the (re)calculation runs, the price records are generated. (This is a change compared to the previous version – price records used to be generated immediately after an agreement was approved.)

Note the following differences between Calculation and Recalculation:

- Calculation on the agreement level does not create any Price Records, it is only a preview of what the results could be, based on the setup. Price Records are generated only when Recalculate is run for the agreement.
- The Recalculation (the periodic one) which is done by Calculation object generates Price Records based on the criteria set in the approved agreement. What gets recalculated is controlled by the feeder logic and the schedule controls how often the feeder is run.

Review Price Records

1. Go to **Agreements & Promotions** > **Price Records**.
2. There can be multiple records for a single product – depending on how many parameters combinations you have defined.
3. What fields are especially useful to check:
   - **Formula Definition Name** – Identifies which formula definition was used.
   - **Formula Type** – Identifies which formula type was used.
   - **Base Adder Type** – Identifies which adder type was defined in the formula definition.
   - **Valid From** / **Valid To** – Represent the defined calculation period (month or quarter).
   - **Output Currency/UOM** – They (together with other fields) distinguish variants of the product price.
   - **Calculated Result** – Result of the calculation with just adders. Bounds are not reflected here.
   - **Engine Result** – Result of the calculation with adders and bounds.
   - **Converted Result** – Reflects the currency conversion (from base currency to output currency). It is calculated as Engine Result times whatever value is in Exchange Rate or Conversion Rate columns.
     Note: The conversion is looked up on the day of recalculation, that is why the results may vary between the agreement and Price Records (if those are generated later on and if the conversion rates changed).

- **Formula Detail** - Shows you details of the calculation.
  For example: *50 + 5 (PRODUCT_EXCEPTION) with 3 MONTH Rolling Period with 1 MONTH Lag Period*
  That is: initial cost + defined adder of 5 (either as percentage or absolute value - as defined) where the price is looked up one month back from the current date (Lag) and averaged using values found in previous 3 months (Rolling Period; this period is moved back by Lag)
- **Calculation Status** - Either OK or Error.
- **Message** - If it says "No cost found", check whether your Product Extension table has cost defined for the given combination of currency and UoM or whether the validity fits the defined range.

Admin User Reference (Agreements)

Installation (Agreements)

⚠ This package is a pre-release version and is not publicly available at PlatformManager. Please contact @Johan Gustavsson before installation.

This section will guide you when installing the Agreements Accelerator package in your environment.

Installation Steps

⚠ Steps below will be applicable once the package is released.

1. Get access to PlatformManager and target partition, as described in common installation prerequisites.
2. Go to PlatformManager at https://platform.pricefx.com/ and log in.
3. Go to **Marketplace** and find the *Agreements* accelerator.
4. Click the accelerator tile, select the partition where you want to deploy the accelerator package, and confirm the deployment dialogue to start.
   ℹ For detailed description of all deployment options, see PlatformManager documentation.

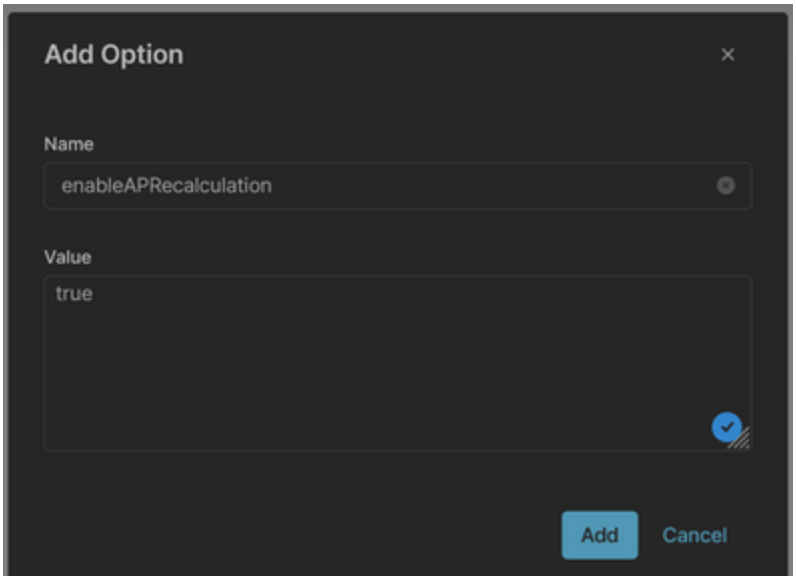
Post-installation Steps

To ensure proper functioning of the Agreements package, you have to go through the steps below. After completing this part, you can start configuring the calculation logics according to your needs.

Enable Price Record Generation

First, you need to enable generation of Price Records for your agreements.

1. Log in to your partition.
2. Go to **Administration** > **Configuration** > **Advanced Configuration Options**.
3. Click the **Add** button at the top right.
4. Enter *enableAPRecalculation* in the **Name** field and *true* in the **Value** field.



5. Click **Add** to save the option.

Enable Agreement Recalculation on Scenario Creation

A recalculation of an agreement is necessary after creating a new scenario. This parameter enables automatic recalculation, so that you do not have to do it manually.

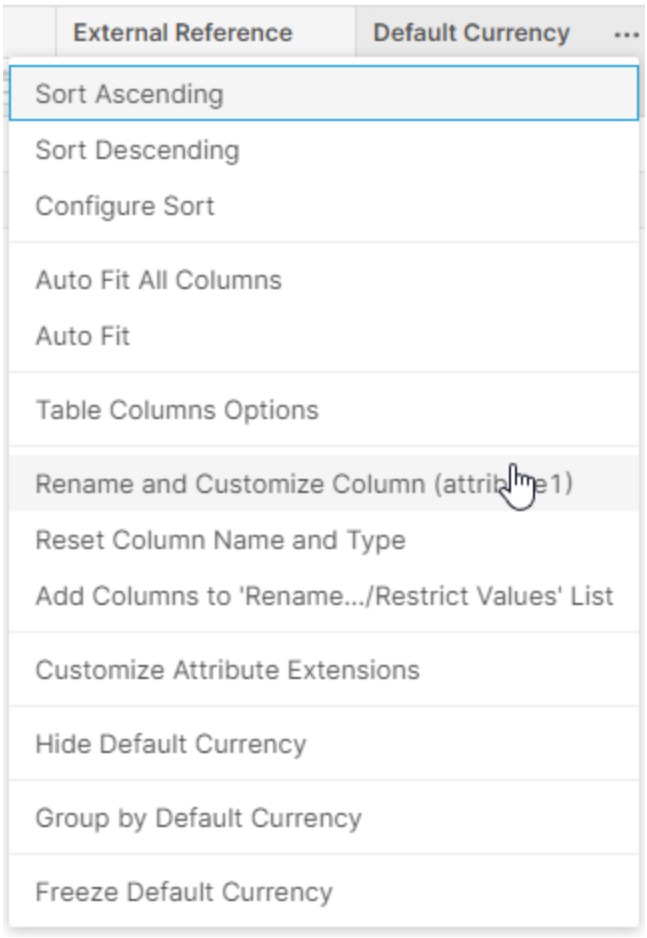
1. Log in to your partition.
2. Go to **Agreements & Promotions** > **Agreements & Promotions Types**.
3. Select the "AGR_FormulaBasedPricing" type and click **Edit**.
4. In the Configuration section, add a new line: `"accScenarioRecalc":"true"`.

5. Click **Save Changes**.

Change Column Labels in Price Records View

To ensure correct functioning of the out-of-the-box Cost Plus formula, you need to change the column names in the Price Record view.

1. Log in to your partition.
2. Go to **Agreements & Promotions** > **Price Records**.
3. To edit a column name, hover over the column and click three dots on the right and select **Rename and Customize Column**.



4. In the popup window, change the name, label and data type for each column according to the data below:
   - "attribute1"
     - name: "defaultCurrency"
     - label: "Default Currency"
     - type: String
   - "attribute2"
     - name: "defaultUOM"
     - label: "Default UOM"
     - type: String
   - "attribute3"
     - name: "calculatedResult"
     - label: "Calculated Result"
     - type: Real
     - format type: Money
   - "attribute4"
     - name: "formulaDefinitionName"
     - label: "Formula Definition Name"
     - type: String
   - "attribute5"
     - name: "validFrom"
     - label: "Valid From"

    - type: String
- "attribute6"
    - name: "validTo"
    - label: "Valid To"
    - type: String
- "attribute7"
    - name: "calculationPeriodType"
    - label: "Calculation Period Type"
    - type: String
- "attribute8"
    - name: "calculationPeriod"
    - label: "Calculation Period"
    - type: String
- "attribute9"
    - name: "engineResult"
    - label: "Engine Result"
    - type: Real
    - format type: Money
- "attribute10"
    - name: "previousResult"
    - label: "Previous Result"
    - type: Real
    - format type: Money
- "attribute11"
    - name: "formulaDetail"
    - label: "Formula Detail"
    - type: String
- "attribute12"
    - name: "exchangeRate"
    - label: "Exchange Rate"
    - type: Real
    - format type: Numeric
- "attribute13"
    - name: "conversionFactor"
    - label: "Conversion Factor"
    - type: Real
    - format type: Numeric
- "attribute14"
    - name: "formulaType"
    - label: "Formula Type"
    - type: String
- "attribute15"
    - name: "baseAdderType"
    - label: "Base Adder Type"
    - type: String
- "attribute16"
    - name: "baseAdderSource"
    - label: "Base Adder Source"
    - type: String
- "attribute17"

        - name: “calculationStatus”
        - label: “Calculation Status”
        - type: String
    - ”attribute18”
        - name: “message”
        - label: “Message”
        - type: String
    - ”attribute19”
        - name: “boundingRuleSource”
        - label: “Bounding Rule Source”
        - type: String
    - ”attribute20”
        - name: “boundingRuleStatus”
        - label: “Bounding Rule Status”
        - type: String
    - ”attribute21”
        - name: “agreementCurrency”
        - label: “Agreement Currency”
        - type: String
    - ”attribute22”
        - name: “agreementUOM”
        - label: “Agreement UOM”
        - type: String
    - ”attribute23”
        - name: “convertedResult”
        - label: “Converted Result”
        - type: Real
        - format type: Money

Technical User Reference (Agreements)

- Company Parameters (Agreements)
- Architecture Diagram (Agreements)
- How to Modify, Add and Remove Inputs
- How to Add New Formula Type with Calculation Method and Inputs
- How to Create New Formula Definition
- How to Modify What Is Saved in Price Records
- How to Add New Outputs to Line Item

Company Parameters (Agreements)

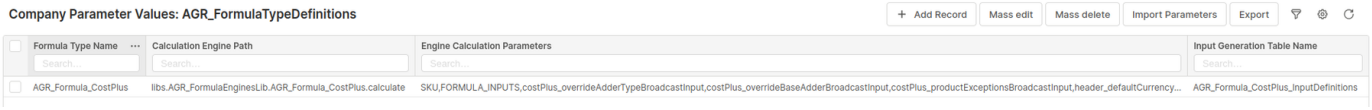
- AGR_FormulaTypeDefinitions
- AGR_Formula_CostPlus_InputDefinitions
- AGR_FormulaBasedPricingHeader_InputDefinitions
- AGR_FormulaBasedPricing_InputDefinitions
- AGR_Formula_ExceptionsConfigurator_InputDefinitions

AGR_FormulaTypeDefinitions

Defines Formula Types and how they calculate the price.

Fields:

- **Formula Type Name** - Stands for the name of an individual Formula Type. These values can then be selected in the **Formula type** dropdown menu in the Formula Definition. Currently, the only type provided out of the box is *Cost Plus*. You can add your own type, for details see How to Add New Formula Type with Calculation Method and Inputs.
- **Calculation Engine Path** - Path to the calculation engine in the format `libs.<name of lib>.<name of element>.<name of method>`.
- **Engine Calculation Parameters** - Parameters to be passed to the calculation. The static ones are: SKU (the product being calculated) and FORMULA_INPUTS (all of the inputs from the Definition step).
- **Input Generation Table Name** - Refers to another Company Parameter table which defines the Input Manager input generation.

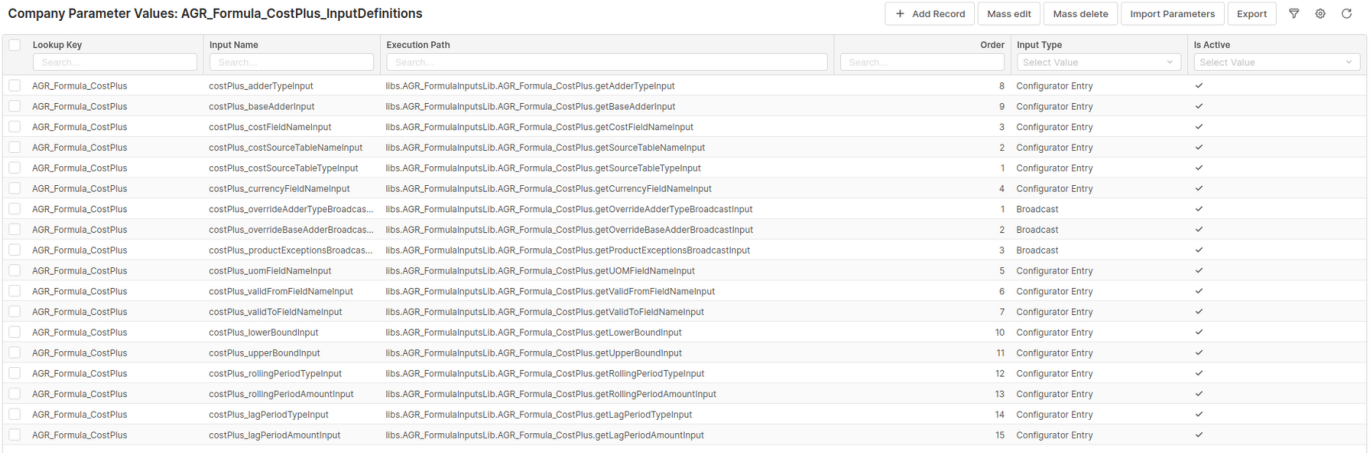


AGR_Formula_CostPlus_InputDefinitions

Allows you to customize the inputs in the Definition step: their order, visibility etc.

⚠️ Do not turn off individual inputs without checking the impact. Some of them are required for the Accelerator to work.

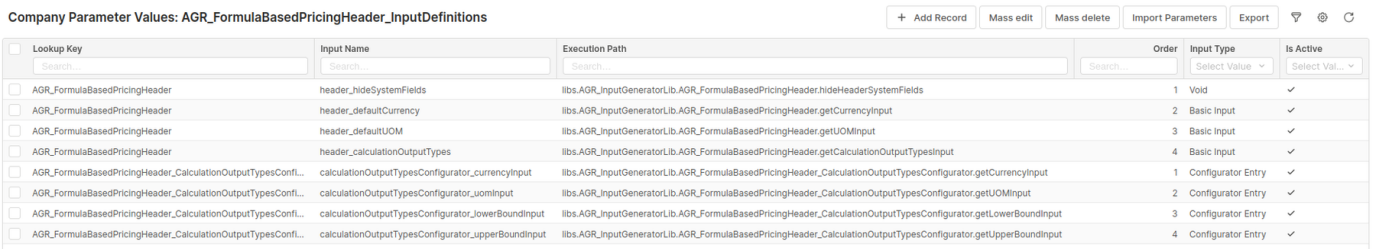
For details on how to add, remove or modify existing inputs see How to Modify, Add and Remove Inputs.



AGR_FormulaBasedPricingHeader_InputDefinitions

Allows you to customize the inputs in the header.

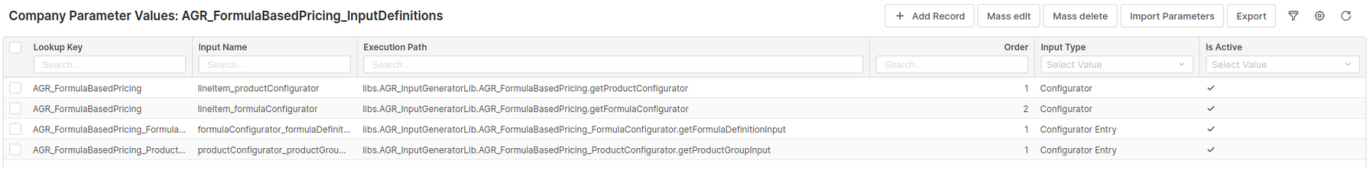
⚠️ Do not turn off individual inputs without checking the impact. Some of them are required for the Accelerator to work.



AGR_FormulaBasedPricing_InputDefinitions

Allows you to customize the inputs on the line item level.

⚠️ Do not turn off individual inputs without checking the impact. Some of them are required for the Accelerator to work.



Company Parameter Values: AGR_FormulaBasedPricing_InputDefinitions

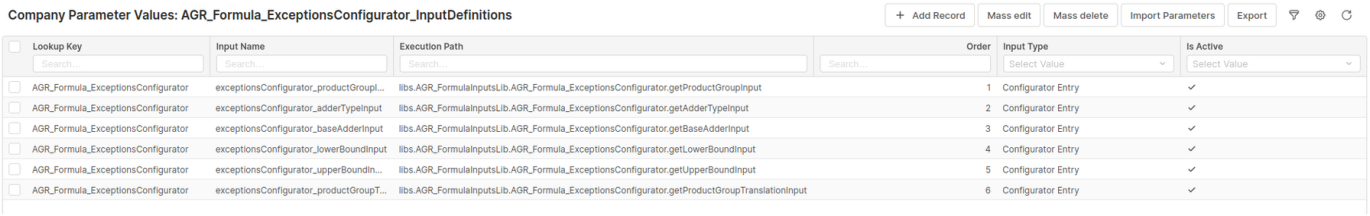
Add Record | Mass edit | Mass delete | Import Parameters | Export

Lookup Key	Input Name	Execution Path	Order	Input Type	Is Active
AGR_FormulaBasedPricing	lineItem_productConfigurator	libs.AGR_InputGeneratorLib.AGR_FormulaBasedPricing.getProductConfigurator	1	Configurator	✓
AGR_FormulaBasedPricing	lineItem_formulaConfigurator	libs.AGR_InputGeneratorLib.AGR_FormulaBasedPricing.getFormulaConfigurator	2	Configurator	✓
AGR_FormulaBasedPricing_Formula...	formulaConfigurator_formulaDefinit...	libs.AGR_InputGeneratorLib.AGR_FormulaBasedPricing_FormulaConfigurator.getFormulaDefinitionInput	1	Configurator Entry	✓
AGR_FormulaBasedPricing_Product...	productConfigurator_productGrou...	libs.AGR_InputGeneratorLib.AGR_FormulaBasedPricing_ProductConfigurator.getProductGroupInput	1	Configurator Entry	✓

AGR_Formula_ExceptionsConfigurator_InputDefinitions

Allows you to customize the exceptions configurator.

⚠️ Do not turn off individual inputs without checking the impact. Some of them are required for the Accelerator to work.

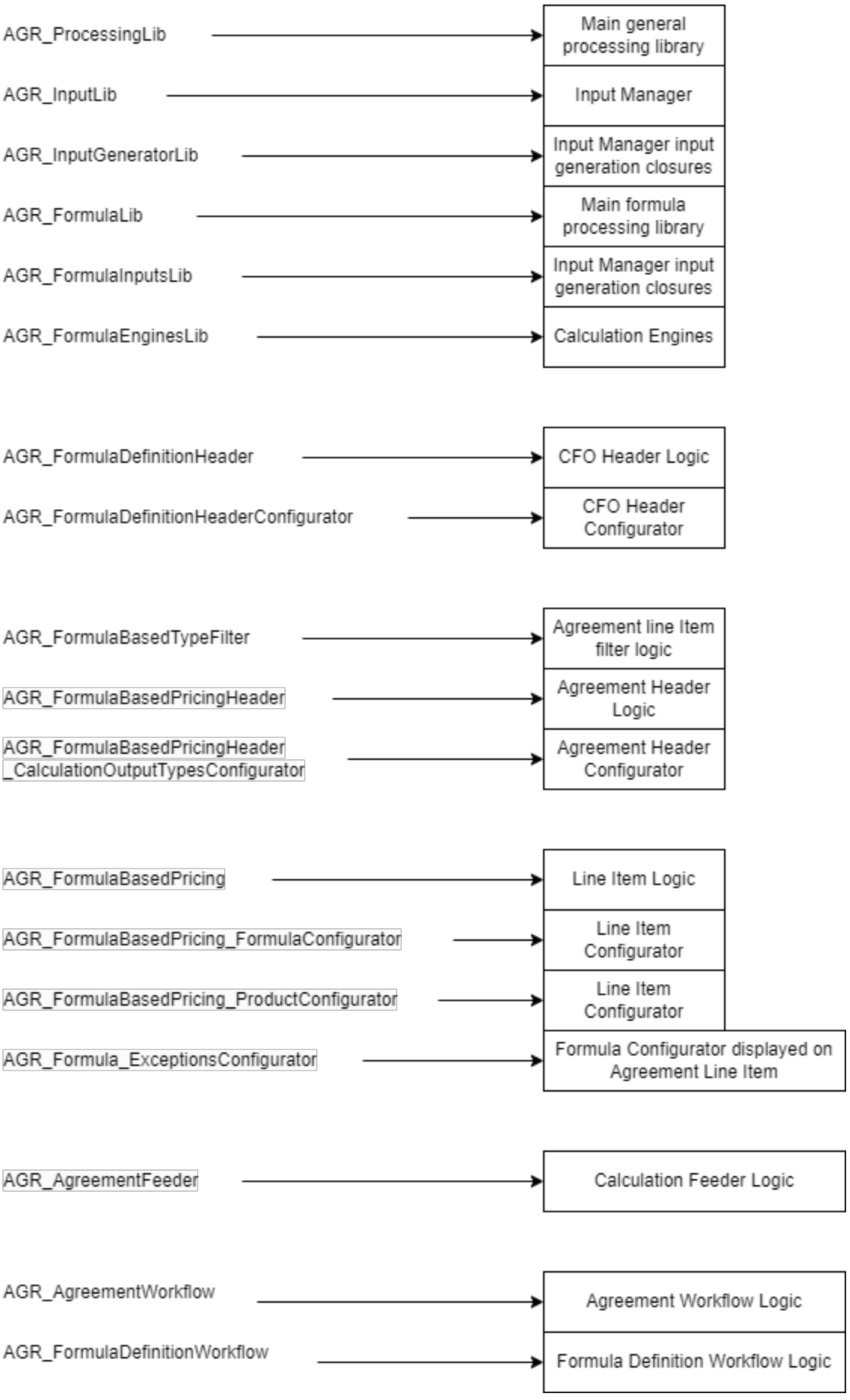


Company Parameter Values: AGR_Formula_ExceptionsConfigurator_InputDefinitions

Add Record | Mass edit | Mass delete | Import Parameters | Export

Lookup Key	Input Name	Execution Path	Order	Input Type	Is Active
AGR_Formula_ExceptionsConfigurator	exceptionsConfigurator_productGroupI...	libs.AGR_FormulaInputsLib.AGR_Formula_ExceptionsConfigurator.getProductGroupInput	1	Configurator Entry	✓
AGR_Formula_ExceptionsConfigurator	exceptionsConfigurator_adderTypeInput	libs.AGR_FormulaInputsLib.AGR_Formula_ExceptionsConfigurator.getAdderTypeInput	2	Configurator Entry	✓
AGR_Formula_ExceptionsConfigurator	exceptionsConfigurator_baseAdderInput	libs.AGR_FormulaInputsLib.AGR_Formula_ExceptionsConfigurator.getBaseAdderInput	3	Configurator Entry	✓
AGR_Formula_ExceptionsConfigurator	exceptionsConfigurator_lowerBoundInput	libs.AGR_FormulaInputsLib.AGR_Formula_ExceptionsConfigurator.getLowerBoundInput	4	Configurator Entry	✓
AGR_Formula_ExceptionsConfigurator	exceptionsConfigurator_upperBoundIn...	libs.AGR_FormulaInputsLib.AGR_Formula_ExceptionsConfigurator.getUpperBoundInput	5	Configurator Entry	✓
AGR_Formula_ExceptionsConfigurator	exceptionsConfigurator_productGroupT...	libs.AGR_FormulaInputsLib.AGR_Formula_ExceptionsConfigurator.getProductGroupTranslationInput	6	Configurator Entry	✓

Architecture Diagram (Agreements)


AGR_ProcessingLib
Main general processing library
AGR_InputLib
Input Manager
AGR_InputGeneratorLib
Input Manager input generation closures
AGR_FormulaLib
Main formula processing library
AGR_FormulaInputsLib
Input Manager input generation closures
AGR_FormulaEnginesLib
Calculation Engines
AGR_FormulaDefinitionHeader
CFO Header Logic
AGR_FormulaDefinitionHeaderConfigurator
CFO Header Configurator
AGR_FormulaBasedTypeFilter
Agreement line Item filter logic
AGR_FormulaBasedPricingHeader
Agreement Header Logic
AGR_FormulaBasedPricingHeader _CalculationOutputTypesConfigurator
Agreement Header Configurator
AGR_FormulaBasedPricing
Line Item Logic
AGR_FormulaBasedPricing_FormulaConfigurator
Line Item Configurator
AGR_FormulaBasedPricing_ProductConfigurator
Line Item Configurator
AGR_Formula_ExceptionsConfigurator
Formula Configurator displayed on Agreement Line Item
AGR_AgreementFeeder
Calculation Feeder Logic
AGR_AgreementWorkflow
Agreement Workflow Logic
AGR_FormulaDefinitionWorkflow
Formula Definition Workflow Logic

How to Modify, Add and Remove Inputs

Most inputs present in the package are handled by InputManager available in *AGR_InputLib* library. InputManager is a solution that enables dynamic management of inputs, reordering, turning on and off (if not required by underlying code) and other functions.

InputManager needs to be initialized in a logic that will use it by calling: `libs.AGR_InputLib.InputManager.getInstance(<CompanyParameterName>, <LookupName>)`

where:

- `CompanyParameterName` is a name of the input generation table defined below.
- `LookupName` is the value of key1 column of that table that defines the set of inputs used by this logic.

Inputs are defined using Company Parameter tables with defined and set structure, which is as follows:

- Table Type: MATRIX
- Value Type: MATRIX2
- Columns:
  - o key1 – Lookup Key
    - This is a key defining the "set" of inputs that will be used during generation.
  - o key2 – Input Name
    - Name of the input. This name can also be used as a calculation parameter for the Calculation Engine. This name must match the name that the input is created with in the code.
  - o attribute1 – Execution Path
    - Generation Path for the input. Provided in format `libs.<name of lib>.<name of element>.<name of method>`. The generation path return value must match the Input Type of the input, defined below.
  - o attribute2 – Order
    - Integer indicating the order of this input compared to others. Inputs are ordered by descending order.
  - o attribute3 – Input Type
    - Type of the input generated by the Execution Path, this is used to handle different contexts, for example Configurators or Broadcasted inputs.
      - Void – the Execution Path returns void, useful for example for hiding system inputs or other cases where the value is not needed.
      - Basic Input – ExecutionPath returns the input generated using `.getInput()` call from InputBuilderFactory.
      - Configurator – ExecutionPath returns the Configurator input Map using `api.inlineConfigurator` or `api.configurator`.
      - ConfiguratorEntry – ExecutionPath returns the ConfiguratorEntry generated using `api.createConfiguratorEntry`.
      - Broadcast – ExecutionPath returns the ConfiguratorEntry generated using `api.createConfiguratorEntry`. This Input Type is used specifically for Formula Types that want to show their inputs on the Agreement level.
  - o attribute4 – Is Active
    - Boolean flag indicating whether the input is shown or not. Please be careful when turning of inputs that may be expected to be present by the underlying code.

After the inputs have been defined, they can be generated in bulk by calling an appropriate method on an instance of the InputManager:

- `getInputs` - Generates all inputs defined as Void, Basic Input, Configurator. Used for example for line item logics, headers, dashboards etc.
- `getConfiguratorEntries` - Generates all inputs defined as ConfiguratorEntry. Used for Configurator logics.
- `getBroadcastConfiguratorEntries` - Generates all inputs defined as Broadcast. Used for Broadcast inputs. This mechanism is already handled by the Accelerator Configurator build for this purpose: *AGR_FormulaBasedPricing_FormulaConfigurator*.

How to Add New Formula Type with Calculation Method and Inputs

To add a new Formula Type with a calculation method and inputs, follow these steps:

- Step 1 - Define Formula Type
- Step 2 - Define Inputs
- Step 3 - Define Calculation

Step 1 - Define Formula Type

Navigate to the AGR_FormulaTypeDefinitions Company Parameter and fill in the columns:

- **Formula Type Name** - Provide a name of the Formula Type that will be displayed in the dropdown in Formula Definition.
- **Calculation Engine Path** - Provide a path to a calculation method in the format `libs.<name of lib>.<name of element>.<name of method>`.
- (Optional) **Input Generation Table Name** - If the Formula Type requires some user inputs to be generated by InputManager, provide a name of another Company Parameter table that will store their definitions.

Step 2 - Define Inputs

Inputs are managed by the InputManager solution available in AGR_InputLib library and described in How to Modify, Add and Remove Inputs.

All inputs used by FormulaTypes must be defined as ConfiguratorEntry (if they are to be displayed in Formula Definition) or as Broadcast (if they are to be displayed in Agreement Line Item).

Keep in mind that the Input Name provided in the InputGeneration table can be used as a calculation parameter that is passed to the CalculationEngine defined in Step 3.

Step 3 - Define Calculation

Calculations are handled by the EngineCalculator solution that is present in AGR_FormulaLib library.

Calculation can have various Calculation Parameters (those are the parameters of the calculation method defined in Calculation Engine Path in the FormulaTypeDefinitions). There are currently two predefined Calculation Parameters:

- SKU - This is the SKU of the product that is being calculated by the Line Item.
- FORMULA_INPUTS - This is a Map containing all inputs present and filled in the Formula Definition that is being used for calculation.

Other Calculation Parameters are input names defined in various places: Header, Scenario (these are the only ones not generated via InputManager yet, their names can be found under SCENARIO_INPUTS field in ConstConfig of AGR_ProcessingLib library), Line Item, Formula Type.

For example, you want to use one of the inputs present in Formula Definition, you can use either the Input Name defined in the Formula Type Input Generation Table or use the predefined FORMULA_INPUTS.

All Calculation Parameters should be entered into the Engine Calculation Parameters column of the AGR_FormulaTypeDefinitions Company Parameter. They should be separated with commas without whitespaces, for example: SKU,FORMULA_INPUTS,myInput_TableName

EngineCalculator requires a certain structure to be returned from your calculation method; the structure can be looked up in the `createEngineResult` method in EngineCalculator and it is as follows:

```
[result        : result,
message        : message,
messageType    : messageType,
additionalInfo : additionalInfo]
```

where:

- `result` key stores the result value (final price) of your Calculation Engine. This value is stored in Price Records in the Engine Result column.
- `message` key can store any message in String format that you wish to return from the engine, it can be a calculation status or something else. EngineCalculator automatically fills this field if it is not overridden.
- `messageType` can store the type of the message provided in the `message` key. EngineCalculator automatically fills this field if it is not overridden.
- `additionalInfo` can store anything else that you wish to return from the calculation. Our CostPlus utilizes this to return a Map with information such as calculatedResult, previousResult, formulaDetail, baseAdderType, baseAdderSource, boundsDataSource, boundsDataStatus all of which are stored in their appropriate columns in Price Records.

Keep in mind that you may need to adjust the Price Records structure to match your calculation and its results.

How to Create New Formula Definition

To create a new Formula Definition to be used in Agreement Line Item calculations:

1. Go to **Agreements & Promotions** > **Formula Definition**.
2. Click the **Create New Formula Definition** button on the top right, provide a label (to identify formula) and other fields if necessary.
3. After navigating into the Formula Definition you get the option to select the Formula Type. Those are defined in the AGR_FormulaTypeDefinitions Company Parameter and described in detail How to Add New Formula Type with Calculation Method and Inputs.
4. After selecting the Formula Type the appropriate set of inputs (if defined in the Formula Type) will be shown. You can fill them with necessary data and then submit the Formula Definition. Before the Formula Definition can be used on an Agreement Line Item it needs to be approved.

How to Modify What Is Saved in Price Records

To modify what is saved in Price Records:

- Step 1 - Modify Price Records Table
- Step 2 - Modify SaveResultToPriceRecords Element
- Step 3 - Modify Price Record Structure

Step 1 - Modify Price Records Table

Go to **Agreements & Promotions** > **Price Records** and modify any columns/attributes to fit your calculation outputs.

Step 2 - Modify SaveResultToPriceRecords Element

Go to the AGR_FormulaBasedPricing logic and open the SaveResultToPriceRecords element.

Then there are several variables that hold important information, those are:

- calculationParameters - Stores all possible parameters to use by Calculation Engine CalculationParameters.
- ProcessedCalculationOutputs - Stores all calculation outputs in Map format. The Map's main key is a combination of Currency and UOM used for that run of calculation (based on selected CalculationOutputTypes) and the content is :

```
[outputCurrency    : outputCurrency,
outputUom          : outputUom,
exchangeRate       : exchangeRate,
conversionFactor   : conversionFactor,
calculationResults : [:]]
```

where:

- `outputCurrency` - Currency of the run as defined by CalculationOutputTypes.
- `outputUom` - Unit of measure of the run as defined by CalculationOutputTypes.
- `exchangeRate` - Exchange rate used to convert from default currency to the current outputCurrency.
- `conversionFactor` - Conversion factor used to convert from default unit of measure to the current outputUom.
- `calculationResults` - Outputs of the EngineCalculator in Map format, grouped by SKU. This contains the map returned by the Calculation method.

Step 3 - Modify Price Record Structure

The current implementation has a certain Price Records structure to fit the Cost Plus use case implemented in the Accelerator. The structure can be freely changed to meet the project requirements.

To change the structure, you must modify the Map variable called *priceRecordStructure* to reflect the changes made in Step 1.

How to Add New Outputs to Line Item

The Agreement Accelerator follows the usual rules when it comes to output generation – that is that each output requires its own element to be present.

To create a new output, create a new element and put it after the CalculationOutputsResultMatrixOutput element and before SaveResultToPriceRecords in the AGR_FormulaBasedPricing line item logic.

Release Notes (Agreements)

Agreements Accelerator 0.1.0

This document summarizes major improvements and fixes introduced in the Agreements Accelerator release version.

Version	0.1.0 (pre-release, available upon request)
Release Date	Jan 29, 2024

Feature List

The following functionalities have been delivered with the initial release of the Agreements Accelerator.

Feature	ID
Agreement Header: Filtering logic for showing relevant condition types for selected header type	PFPCS-7277
Agreement Header: Input Parameters	PFPCS-6897
Agreement Line Item: Add new Product Configurator	PFPCS-7735
Agreement Line Item: Outputs	PFPCS-7150
Agreement Recalculation: Feeder logic for picking contracts to be recalculated	PFPCS-7276
Agreement Recalculation: Set up scheduled recalculation	PFPCS-7508
Formula Library: Implement AWP for Formula Definitions	PFPCS-7734

Formula Library: Introduce "Cost +" Formula Elements & Definitions	PFPCS-7085
Formula Library: Introduce the Formula Library	PFPCS-7083
Input Manager: Introduce extendable inputs to Contract objects	PFPCS-7124
Price Record: Record Structure	PFPCS-7588
Scenario: Folder definition rename	PFPCS-7630
Scenario: Inputs	PFPCS-7247