Accelerate Price Flexibility Package

Version 1.2.0

July 2021

Accelerate Price Flexibility Package

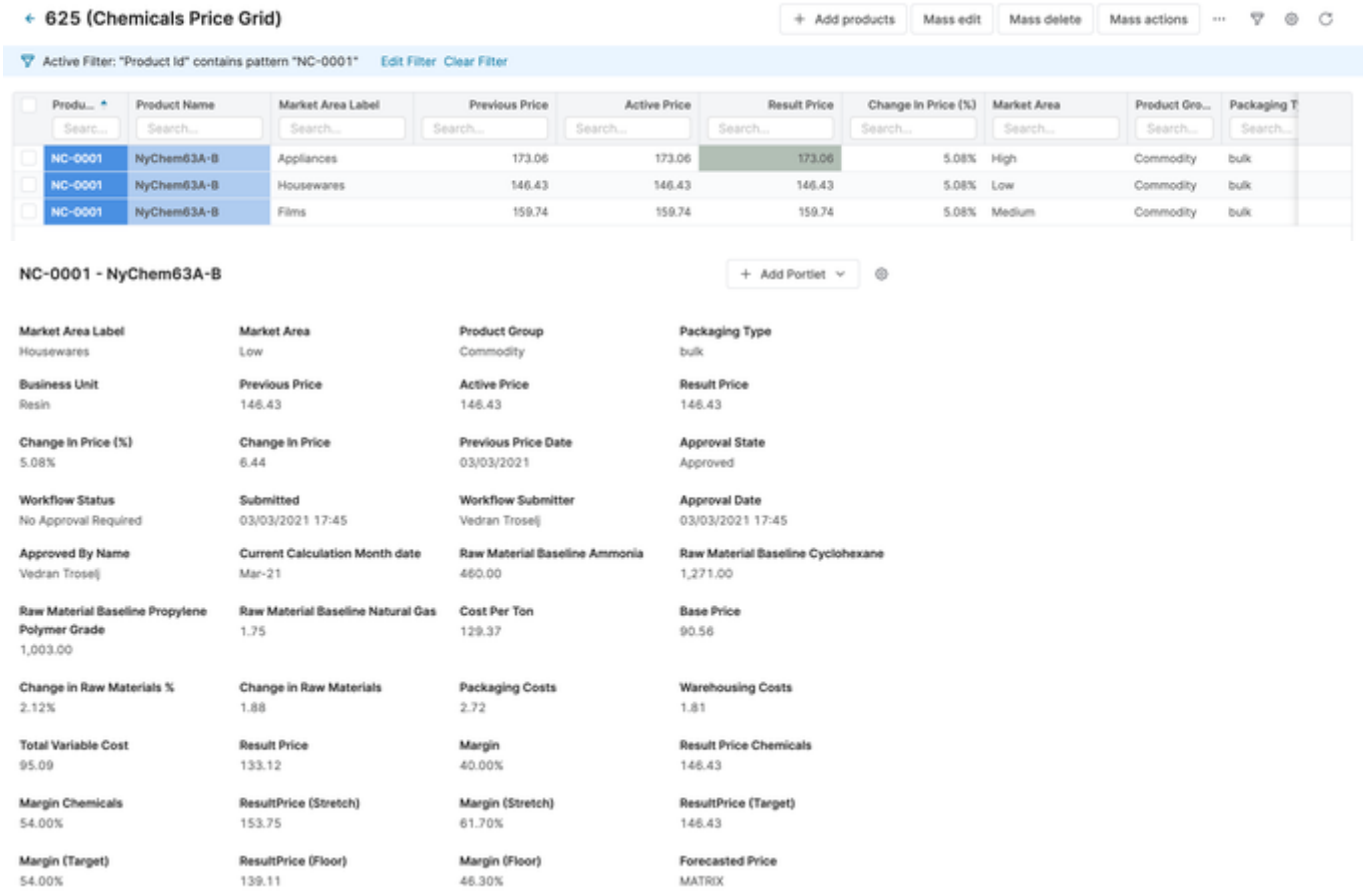
The Price Flexibility Package ensures ongoing price maintenance and updates:

- Price recalculation is triggered if there are changes (e.g. new cost, new information on a competitor price) or new products added.
- A ToDo item can be created automatically and products are added to an LPG.
- After the calculation, the price can be approved.
- Finally, the product is updated in an existing Price List (or added if it is a new product).

Key Deliverables

With this package you can increase your flexibility for rapid market changes through our dynamic pricing capabilities. You can also automate your regular price maintenance and updates through price recalculations based on defined triggers and formulas.

- **Notifications** related to newly added / updated products
- Definition of **rules for price building logic** and automated price updates through Live Price Grids
- **Scheduling** of automated pricing updates
- Automated invoking of **new pricing approval process** through automated workflows
- **Deploying new pricing** into general price lists

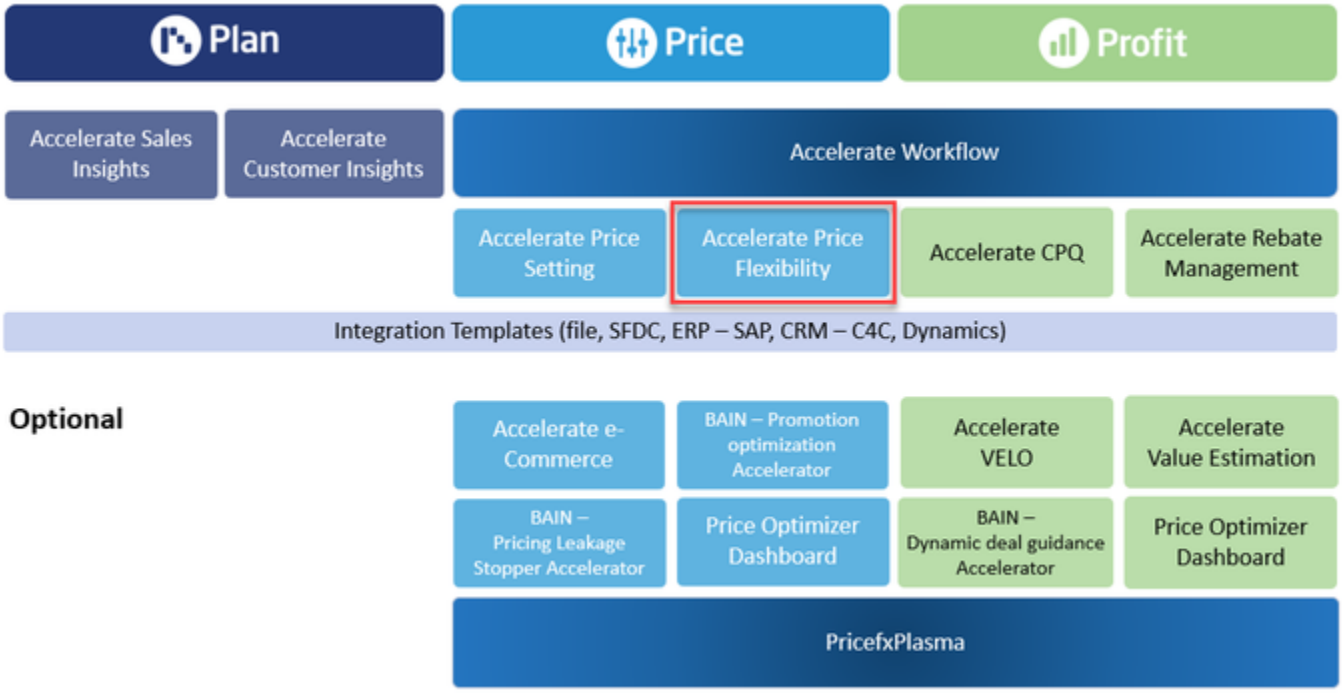


Further reading:

- Overview (Price Flexibility)
- Admin User Reference (Price Flexibility)
- Release Notes (Price Flexibility)
- Archive of Documentation (PFP)

Overview (Price Flexibility)

The Price Flexibility Accelerator is one of many pre-built solutions from Pricefx that when implemented will increase your nimbleness for adjusting to rapidly changing market conditions using the Pricefx dynamic pricing capabilities. It will allow for automation of price maintenance and recalculations based upon established triggers (material costs, competitor pricing, etc) and defined formulas.



Within the design of the Pricefx PLAN/PRICE/PROFIT structure, we can see exactly where the Price Flexibility Accelerator will be included:

In this section:

Business Overview (Price Flexibility)

Premise

You need to manage the maintenance and updating of existing price lists in response to price triggers as part of a team from either Pricing, Finance, Sales, or Sales Ops within your organization.

Desired Outcome

Increase your flexibility for rapid market changes through our dynamic pricing capabilities. Automate your regular price maintenance and updates through price recalculations based on defined triggers and formulas.

Context and Background

The organization has an existing set of price lists that need to be administered and maintained in response to market changes. Your current pricing application doesn't provide a framework that performs automated recalculations using standard business rules that generate updated prices in a timely fashion.

Problem

The need for the definition of standard pricing rules for the automation of product price recalculations that are either triggered by an event (material cost increase, competitor price change) or happen on a regular schedule. Additionally, these price changes will proceed through an automated approval process to increase the speed of their deployment.

Solution Capabilities

Once this Accelerator has been implemented and linked to your application, these components will be available for immediate use:

- Ability to define rules for automated price updates
- Deployment of dynamic pricing through Live Price Grids
- Scheduling of automated pricing updates
- Creation of approval workflows for price updates
- Notifications of new or update products

Capabilities Summary (Price Flexibility)

Capabilities

- product price recalculation based on defined triggers
- available triggers
  - new product added
  - product update (generally a change of product attribute values) - either in table Products or Product Extension. Examples of changes:
    - product Life Cycle state
    - new color for a product
    - new cost
    - new competitor information

Limitations:

- the destination Pricelist (where price changes are finally stored) should not be used for recalculation, only as storage of the prices

How it works

1. changes in data detected
2. product added to special LPG, and recalculated
3. "TODO" item created for a user so they know, they have to approve the price modifications
4. new price is approved/denied in special LPG (by users or automatically based on rules). Note, that this is not a feature of the Price Flexibility accelerator.
5. approved product price is deployed/updated in existing Pricelist
6. product is removed from the special LPG

Audience

Technical Admin

installation and initial setup of the accelerator

Business Admin

(likely Pricing Manager) setup of the triggers, which trigger recalculation

Business User

(likely Pricing Managers) receives the TODO notifications, approves/denies the new prices

User Stories

Automated Price recalculation

I want to

Recalculate Products based on specified conditions (e.g., new cost, change of Product lifecycle state, new Product, Product with a new color, etc.).

so I can

Ensure up to date prices and reduce price stagnation

New Price verification

I want to

Create a ToDo for people responsible to verify the change of Price.

so I can

Bring attention to the responsible/designated approver to review price changes

Price List Refresh

I want to

Update of relevant Price List with the newly approved price(s).

so I can

Allow approved price items to be published and denied price items to stay without a price update

Admin User Reference (Price Flexibility)

- How Price Flexibility Package Works
- Price Flexibility Overview
- Configuration (Price Flexibility)
- Architecture Components (Price Flexibility)

How Price Flexibility Package Works

This section explains the technical background of the Price Flexibility Package cycle. It consists of the following steps:

- Detect Change
- Add Product
- Set Schedule
- Define LPGs
- Move Product to LPG/PL
- Assign ToDo Item

Detect Change

Changes can be detected only for the dependency level context. You can read about dependency levels concepts in Price Setting Package documentation in DependencyMappingConfig PP, DependencyConfiguration PP and Dependent Price Lists and Data Fallbacks. When a change is detected, it is registered in LifecycleMonitor PX and one of the following happens:

- A new product was added to the system.
  - A flag "New Product" is set in the LifecycleMonitor PX in the Lifecycle Status column.
  - A flag "Needs Recalculation" is set in the LifecycleMonitor PX in the Calculation Status column.
- An observable attribute of an existing product was changed.
  - A flag "Needs Recalculation" is set in the LifecycleMonitor PX in the Calculation Status column.
  - A new value is set to the appropriate column of the LifecycleMonitor PX. These columns are defined by the EnrichMonitorPP CF and their indexes are stored in the ObservableAttributes PP.
  - A change is detected if there is a pair of old and new values for any observed attribute.

- If an observable attribute of some product does not have a value, a placeholder <<null>> value is used to differentiate between an empty value and a missing value.

Add Product

Every changed product is added as a new element to LPGs defined in the PriceGridMapping PP. Price grids do not accept duplicate elements, so if a product is already there, nothing will happen, but the price will be changed accordingly on next recalculation.

Set Schedule

Products are added to the configured LPGs automatically after the scanning interval passes. By default it is every 24 hours. It can be changed by modifying the schedule in the DetectChangesSchedule PP.

Define LPGs

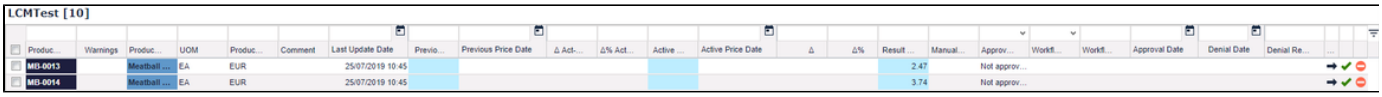
The number of LPGs may vary. They are configurable in the PriceGridMapping PP in the LPG Id column.

This is what the sample configuration could look like:

It is possible to to leave the Price List Id column entry empty. It will stop the process for a given LPG Id, so products will not be moved to a PL when approved and they will not be deleted from Price Grid when denied.

Example:

Every LPG can have its own calculation logic to receive their own result prices.



Move Product to LPG/PL

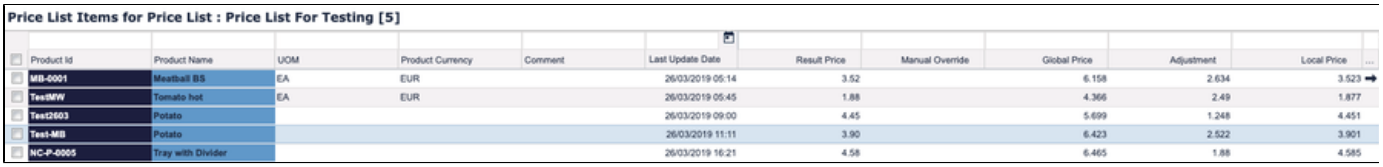
After a new product was approved in a LPG (when the user clicks the approve icon or if it is auto-approved by the assigned workflow), this price record is moved to the Price List which is configured in the PriceGrid Mapping PP in the Price List Id column. If any extended attribute (attribute1, attribute2, etc.) in the source LPG has a custom name and there is an attribute with the same custom name defined in the target Price List, the value will be copied from the source attribute to the correct attribute of the target.

> ⚠ Because of some API restrictions, we cannot copy all information that is available in the "Details" popup of PL/LPG. To avoid mismatch in the data, the target PL should not be used for price calculation – only for price storage, as the source LPG is already used for price calculations.

When the product information is moved to the target PL, the information about the old observed product attribute values is cleared from the LifecycleMonitor PX.

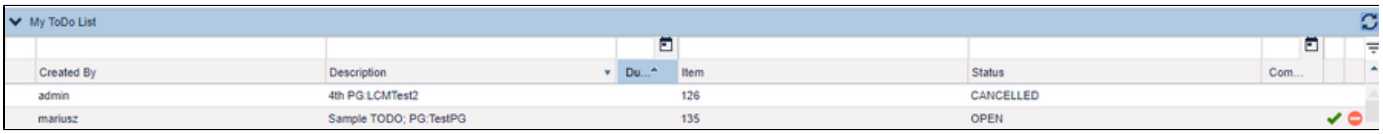
If the new product was denied by the user, it is removed from the LPG automatically and the information about the old observed product attribute values is cleared from the Lifecycle Monitor PX.

These operations are handled by the ProcessApprovalProductInMonitor CF. By default - every 1 hour.



Assign ToDo Item

"My ToDo List" will appear on the Classic UI homepage. The status now is OPEN. The user can approve or deny this task but it will be auto-approved once all elements are approved or denied in the LPG. It will be reopened when new items are added to the LPG, so no duplicate ToDos will be created.



Price Flexibility Overview

This package is designed to notify a user group to follow the status when a product is added or updated. It means that changed products are added to a Live Price Grid where the change is waiting to be approved or denied. Once the decision is made by a member of the assigned user group, approved products are moved to a configured Price List and denied products are removed from the Live Price Grid.

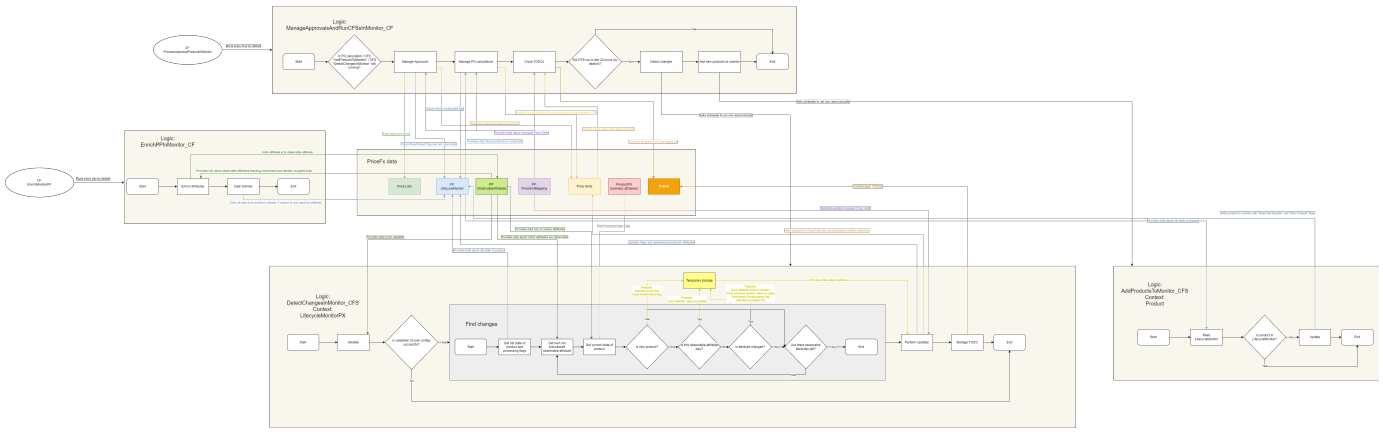
Price Flexibility supports up to 24 observable attributes. Each of them is a product property in either Product or PX table.

Price Flexibility package monitors all products available in the system; currently it is not possible to filter them in any way.

Products are added to the Lifecycle Monitor when:

1. They are new in the Product Master.
2. Once a value of a configured observable field has been updated.

Observed fields can come from Product Master or any Product Extension. ProductExtension, DependencyLevel and Fallback Mechanism known from PSP are supported. You can read more about fallbacks in the PSP fallback section.



Configuration (Price Flexibility)

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

- Price Flexibility Deployment
- Price Flexibility Setup
- Price Flexibility Configuration Price Parameters
- Price Flexibility Advanced Technical Information

Price Flexibility Deployment

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 general steps described in the PlatformManager documentation.

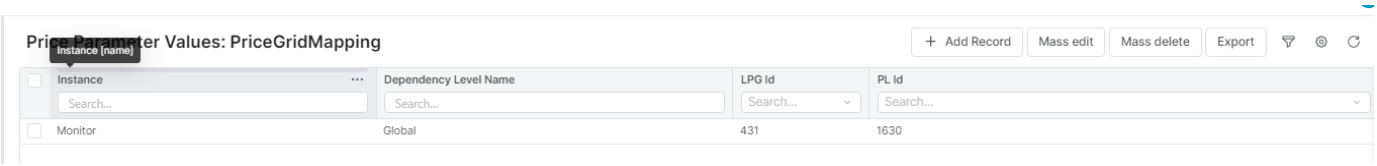
Price Flexibility Setup

Before setting up Price Flexibility, you should read Monitor Instances and Dependency Configuration to understand concepts of monitor instances. Then, to make the package work properly, the following steps have to be done after deployment:

- Define Monitor Instances and Price Grid Mapping
- Add Observable Attributes
- TODOConfiguration
- First Run

Define Monitor Instances and Price Grid Mapping

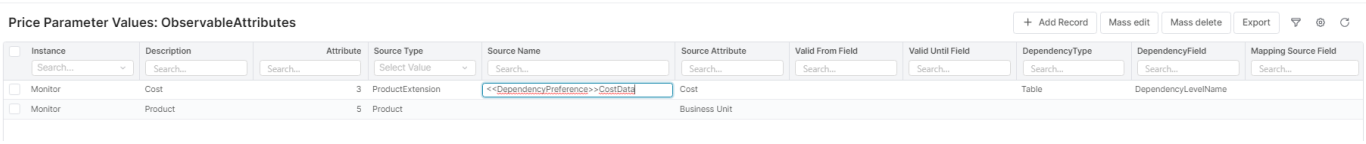
In the PriceGridMapping PP you should define outputs of LifeCycle Monitor and monitor instances:



Please keep in mind that PL Id is optional, it is not required for minimal version of Price Flexibility.

Add Observable Attributes

In ObservableAttributes PP you should configure which changes of a product will trigger LifeCycle Monitor.



Please note that the last 5 attributes are fully optional.

Currently observable attributes are not actually optional. If there are 0 attributes in the PP table, an error will be thrown. At least 1 observable attribute needs to be added with current logics.

Moreover, the "Attribute" column should be left empty (it will be added by CF which is run daily) and "Source Name" is valid only in case of ProductExtension Source Type. Description is a name of your choice, to differentiate easily between parameter names.

Make sure that after changing the ObservableAttributes PP the EnrichMonitorPP CF is run.

TODOConfiguration

TODOConfiguration PP defines for whom to create TODO tickets from which PG. This step is optional.

First Run

- Remember that CF runs CFSs in parallel. Even when you see a finished job, other CFS might still be working.
- CF runs every hour by default. However, CFS is run by CF every 1 day (~ every 24 runs of CF).
- After the first run, verify that all products were added to the LifecycleMonitor PX.

From now on, changes on observable attributes will be monitored and processed using this package.

Price Flexibility Configuration Price Parameters

The configuration options described here may be required to set up individual Price Flexibility features.

- ObservableAttributes PP
- PriceGridMapping PP
- TODOConfiguration PP
- DetectChangesSchedule PP
- PF_DependencyConfiguration PP

ObservableAttributes PP

Column name	Instance	Description	Attribute	Source Type	Source Name	Source Attribute	Valid From Field	Valid Until Field	Dependency Type	Dependency Field	Mapping Source Field
Allowed Values			Filled by the system	• Product • ProductExtension					▪ Table ▪ Lookup		
Description	Used for mapping to PriceGridMapping for detection (Dependenc	Unique business name of the	Do not touch if not instr	Source type for data	Name of the source. Leave empty for the	Attribute name in the source table,	Attribute name in the source table used for the time constraint (optional), e.g. "ValidFrom". Value	Attribute name in the source table used for the time constraint (optional), e.g. "ValidTo".	Dependency mapping mechanism is done in two	Name of th	Used onl

	yMapping) and managing approvals.	observed attribute.	ucted.		source type "Product".	e.g. "ProductCost"	s in this column must be date time and they indicate from which date the entry is valid.	Values in this column must be date time and they indicate until which date the entry is valid. If no "Valid Until Field" is provided, a valid entry is the one with the latest past date in the "Valid From Field" column.	ways (or leave it empty): • **Lookup** – Assumes that you have a data source for all countries. E.g. one PX for the product cost for different products and values 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	ecolumninPF_DependencyConfiguration(orDependencyConfigurationfromPSP)formappingdatatode	yinLookuptypeofDependencyMapping.Nameofthecolumninsourcetableformappingdependencyle

									sources for all dependencies. Each dependency has its own data source for a specified dependency mapping mechanism. When you use this type of search, you should also change the value of sourceTable in PriceSettingConfig: the value should include a placeholder that will be swapped with our depe	pendencylevel	vel to data.

									ndency 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. Table dependency works for PX only, because		

Pricefx has only 1 P table. An example of usage:

- DependencyLevelName: Germany
- Preference1 (fromPF_DependencyCo

nfiguration): DE
- Dependency Field: Preference1
- SourceType: PX
- SourceName: ProductCosts

<<DependencyPreference>>Withthisdata,thedependencymappingmechanismwillsea

									rch for a PX named Product Costs DE and then perform the lookup.

⚠ Notes:

- Currently observable attributes are not actually optional. If there are 0 attributes in the PP table, an error will be thrown. At least 1 observable attribute needs to be added with current logics.
- There is a hard limit of 24 observed attributes. Additional attributes will not be observed. The trade-off is that adding a new attribute in place of the deleted one will be more time consuming. There is a lazy deletion done by EnrichMonitorPP CF and values of no longer observed columns are cleared when a new space is needed.
- When you add a new observable attribute, the system needs to assign a free attribute pair to it. This is done by the "EnrichMonitorPP" CF. It is scheduled to run once per day. You also can start an immediate execution when you want to have on demand initialization.

- Do not create new rows by copying existing ones because you will also copy the read-only "Attribute" value. It is important to leave this column empty as the proper "Attribute" number should be assigned automatically by EnrichMonitorPP CFS.

PriceGridMapping PP

Column name	Instance	Dependency Level Name	LPG Id	Price List Id
Allowed Values				
Description	Used for mapping to observable attributed for detection (DependencyMapping) and managing approvals.	Name of the DependencyLevel from PF_Dependency Configuration, or Dependency Configuration.	LPG into which products with detected changes will be added.	Optional field. Defines a Price List where approved products will be moved. If not defined, items will stay in the LPG regardless of their approval status.

This price parameter can have multiple rows. It means that detected changes can add products to multiple LPGs and they will be moved to corresponding PLs. Because rows are handled by a single LifecycleMonitor PX, all changes will be visible on all LPGs.

These configs are not relevant and will be ignored if they do not have a corresponding observable attribute in the Observable Attributes PP. The whole PF works from the context of observable attributes mapped with PriceGridMapping by monitor names.

To avoid race conditions and counterintuitive behavior, we discourage using multiple monitors with same dependency level or multiple monitors with same target.

TODOConfiguration PP

Column name	Monitor instance name	User group name	TODO description
Allowed Values			
Description	Name of the monitor from the PriceGridMapping PP.	User or group that a given TODO will be assigned to.	Description of the TODO visible for the user.

Monitor name and Group are both keys which means that any combination of these two can be created for a new TODO.

DetectChangesSchedule PP

Name (hardcoded)	

	Value (user defined / technical).
Detect Changes Every [min]	Defines how often the detect changes job should run.
Last Execution Timestamp	Technical field. Timestamp of the last execution. Do not change if not instructed.

⚠ The CF ProcessApprovalProductInMonitor is the "heartbeat" of the scheduling. It runs every 60 minutes by default. If you need a more frequent trigger, you can reduce it to 5 minutes in the CF scheduling configuration. It is not recommended to reduce it below 5 minutes due to technical restrictions. With the Detect Changes Every Parameter you can "skip" execution of this task and reduce it to e.g. once every 24 hours.

PF_DependencyConfiguration PP

Column Name	Dependency Level Name	Depends On	Dimension	Currency	IsComplete	Preference #01-27
Values	{name of dependency}	{master level name}	Data used to help describe what the dependency is used for. If one level depends on another, that relation will be treated as HQ mode.	{currency code}	Flag indicating if all data is supposed to be filled at this point. If the dependency level contains "Yes", fallback hierarchy will be stopped at this point.	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"	e.g. "Country", "Warehouse"	e.g. "EUR"	Allowed values: • Yes • No	e.g. "ISO Code", "SalesOrg"

⚠ This Price Parameter is optional. If users already use PriceSettingPackage, they can leave this PP empty. Then the DependencyConfiguration PP will be used. DependencyConfiguration from PSP has 4 keys, 2 of them will not be utilized.

Price Flexibility Advanced Technical Information

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

Some of 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.

- Shared Cache Usage
- Monitor Instances and Dependency Configuration

Shared Cache Usage

Due to optimization reasons, changes detecting uses Shared Cache. When the DetectChanges CFS runs right after the previous instance has shut down, PP configuration will be not read again (it will be the same as before). Shared Cache is cleared after 15 minutes of not accessing it, so it should be at least that much time between different DetectChanges instances runs.

Monitor Instances and Dependency Configuration

In Price Flexibility Package you can configure multiple monitors. Each monitor can have its own:

- Price Grid Mapping (which PG product with detected changes is monitored)
- Observable Attributes (which changes of a product will trigger a LifeCycle Monitor)
- Dependency Level

Dependency Levels require most attention. They are responsible for multi-level fallback. Each piece of data can be duplicated for each Dependency Level to simulate, for example, different costs of production among countries. However, "Plus" in "Cost Plus" strategy might be different per vendor, not per country. Because of that, we can keep data on different Dependency Level. The lookup will be performed for more and more generic levels, until data is found.

Hierarchy used will be from DependencyConfiguration PP if PSP is deployed, or https://pricefx.atlassian.net/wiki/spaces/ACCDEV/pages/3377529200/PF+DependencyConfiguration+PP if no PSP is deployed.

Please note that, unlike in PSP, users do not need to use Dependency Mapping. Then the configuration should be left empty.

Restrictions of Multiple Monitor Instances

Currently, these are downsides of using this feature:

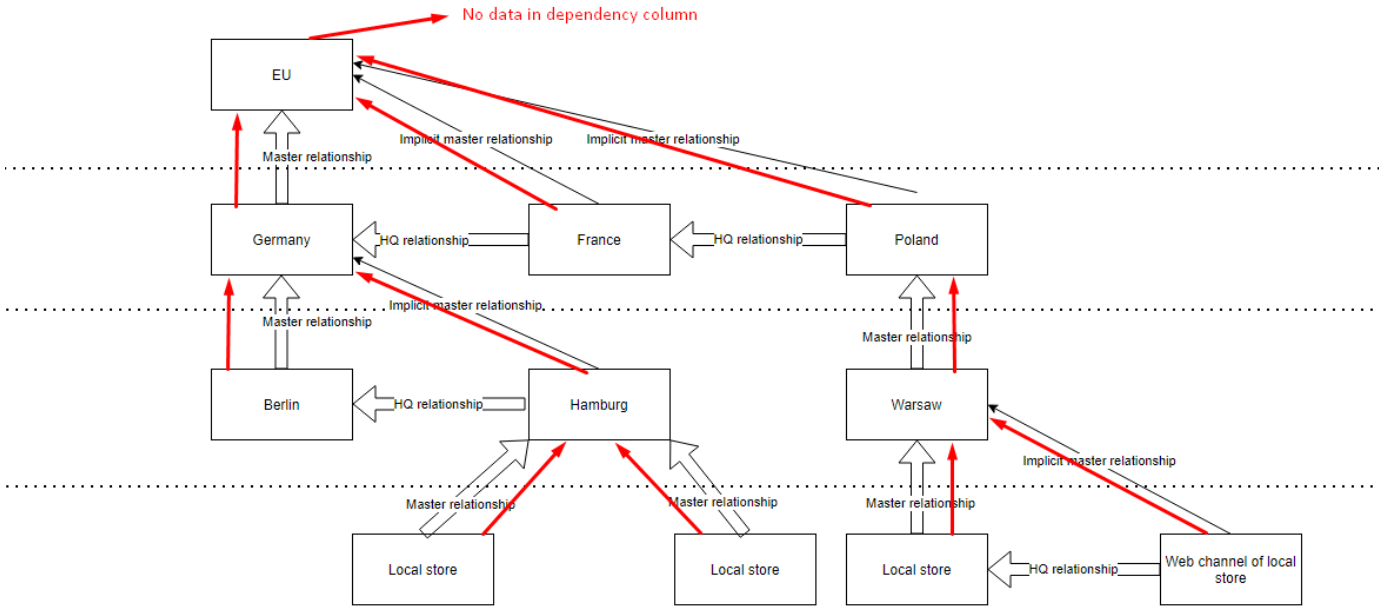
- Price Flexibility Package Price Setting Package integration is supported for only 1 monitor instance.
- 'Needs Recalculate' flag (which is an optimization trick) is utilized properly for only 1 monitor instance.

Fallback Rules

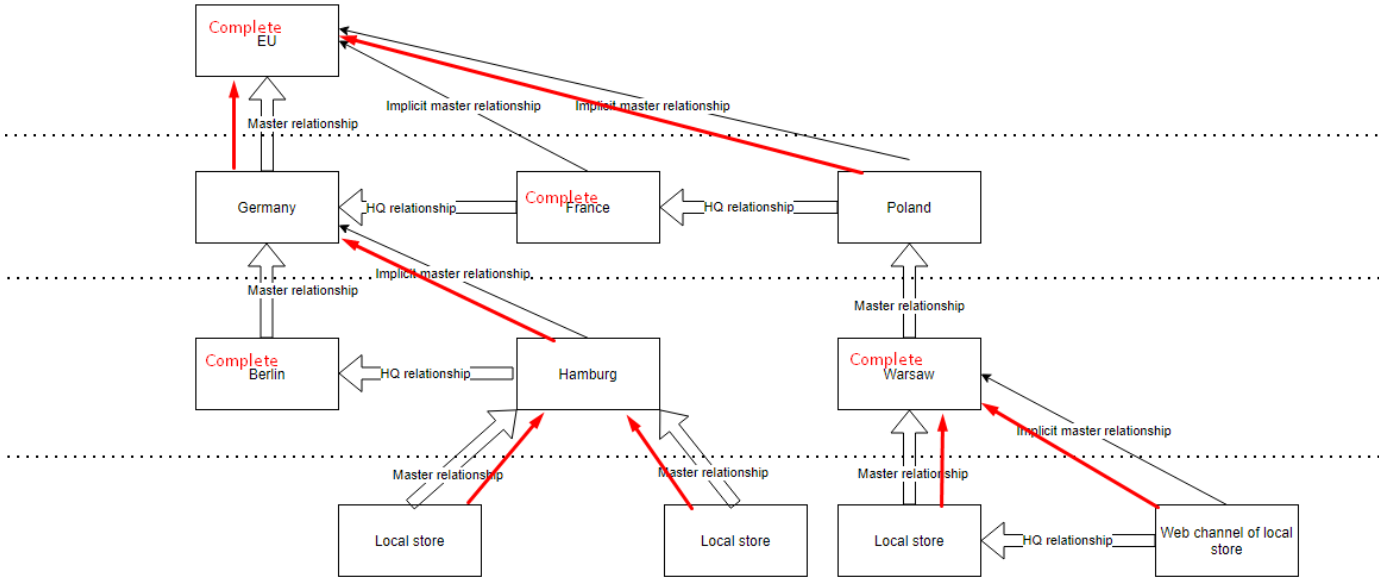
Fallbacks follow these rules:

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

Example without the isComplete flags:



Example with the isComplete flags:



Architecture Components (Price Flexibility)

Calculation Field Set

- AddProductsToMonitor
  - logic AddProductsToMonitor_CFS
  - table Products
- DetectChangesInMonitor
  - logic DetectChangesInMonitor_CFS

- table PX LifecycleMonitor

Calculation Flow

- EnrichMonitorPP
  - logic EnrichPPInMonitor_CF
- ProcessApprovalProductInMonitor
  - logic ManageApprovalsAndRunCFSsInMonitor_CF

Logics

Generic

- AddProductsToMonitor_CFS
- DetectChangesInMonitor_CFS

Library

- LifeCycleLib
- MonitorConfigurationManager

Calculation Flow

- ManageApprovalsAndRunCFSsInMonitor_CF
- EnrichPPInMonitor_CF

Customer Parameters

- DetectChangesSchedule - including data
- PriceGridMapping - data can be uploaded by user already during installation
- TODOConfiguration - data can be uploaded by user already during installation
- ObservableAttributes - data can be uploaded by user already during installation

Preferences

Intended for the company parameters field layouts.

Product Extension

- LifecycleMonitor

Dependencies

This accelerator depends on these accelerators which will be deployed during the installation too:

- Shared Library

Price Flexibility Package 1.2.0

Bugs

PFPCS-4096 Caught the error in DetectChangesInMonitor CFS logic

Change Requests

PFPCS-658 Add Dependency Mapping mechanism to PF

Stories

PFPCS-3605 Add validation of observable attributes configuration

PFPCS-3269 Change Element to "Displaymode = Never" in all CFS

PFPCS-1895 Multiple monitors in price flexibility package

- **Upgrade notes**:
  - Ensure that CFS:
    - AddProductsInMonitor is unique and doesn't have targetDate
    - DetectChangesInMonitor is unique and doesn't have targetDate
  - Changes in PP PriceGridMapping:
    - Columns moved:
      - LPGId moved from name to attribute2
      - PLId moved from attribute1 to attribute 3
    - New columns:
      - name: Instance
        - Unique name of the monitor. Monitors are entities with their own observable attributes and dependency levels.
          - Each mapping has its own monitor.
          - Each monitor has only one dependency level defined.
          - Each monitor might have multiple observable attributes.
      - attribute1: DependencyLevelName
  - Changes in PP TODOConfiguration
    - key1
      - Column name: "monitorName"
      - Content: Unique name of the monitor.
  - New PriceParameter, PF_DependencyConfiguration. It is used for hierarchical lookups.

    - User might use DependencyConfiguration from PSP. Create PF_DependencyConfiguration and leave it empty for that
    - PF_DependencyConfiguration should be a matrix type with 2 keys:
        - key1: DependencyLevelName
        - key2: DependsOn
    - Attributes:
        - attribute1: Dimension
        - attribute2: Currency
        - attribute3: IsComplete
        - attribute4-30: Any user preferences, used for mapping dependency levels to data
- Changes in PP ObservableAttributes:
    - Type changed from matrix with 1 key to matrix with 2 keys.
        - key1: Instance
            - Name of the monitor. Might be not unique, allowed values from PriceGridMapping entries.
        - key2: Description
            - Data from former “name” column
    - Three new attributes for DependencyMapping:
        - attribute7: DependencyType
        - attribute8: DependencyField
        - attribute9: MappingSourceField
- How to keep old behavior:
    - Create new monitors with unique names.
    - Keep Dependency Level Name and DependencyMapping fields empty.

Tasks

PFPCS-4078 PF - Setup minimum version number and documentation URL in PM