Accelerate Sales Compensation

Version 1.4.0

October 2022

Accelerate Sales Compensation (SC)

The Sales Compensation Accelerator provides a solution for fast implementation of the company's sales compensation plan.

The package includes:

- Common compensation types
- Approval workflow for compensation agreements, agreement records and payout values
- Output templates for documents like Sales Compensation Agreement, configuration overview and progress reports
- Dashboards with information about progress towards the plan
- Dedicated views for compensation agreements and agreement records

Search for configuration, manuals or reference information.

Search

In this section:

- Product Info (SC)
- Get Started (SC)
- Reference (SC)
- Technical Information (SC)
- Release Notes (SC)
- Archive of Documentation (SC)

Product Info (SC)

The Sales Compensation Package provides a solution for the fast implementation of the company's sales compensation plan. Its purpose is to maintain, control and evaluate execution with the Pricefx application.

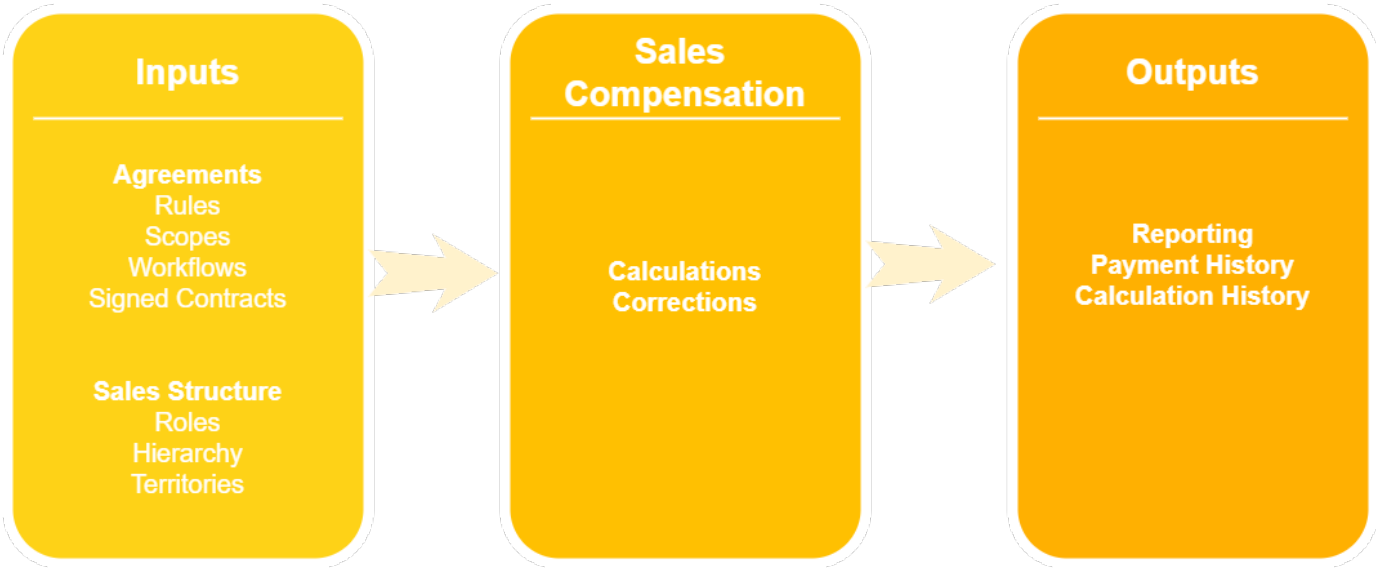
In this section you will find:

- Sale Compensation Plan
- Business Roles
  - Sales Compensation Administrator
  - Sales Manager
  - Sales Agent
- Scope of Usage
  - Creating Compensation Agreement
  - Reporting Progress
  - Plan Maintenance
- Reporting and Dashboards
  - Compensation Dashboard
  - Overview Report
  - YTD Report
  - Transaction List

Sale Compensation Plan

It outlines the sales agent's base salary as well as the company's commission and incentive program with strategy. Commission, bonus, and incentive structure incentivize sales force to reach their objectives in order to earn a deserved reward. Final pay mix together makes up On-Target Earnings. In other words, the amount of money a salesperson is paid in the end.

Sales compensations encourage positive behaviors in your sales teams that are necessary to achieve your overall organizational goals and results.



Business Roles

The following business roles and individuals participate in the processes surrounding the Accelerator's usage activities.

Sales Compensation Administrator

A person who is responsible for the smooth working of all elements. This person takes the plan from the company's representatives and ensures that individual Sales Managers can create agreements. The person makes sure all salespeople are in the system and individual compensation types and calculation logics that the company uses are present in the system. Sales team changes, adjustments or running plans, contract revisions for a new cycle, or global reporting are activities in their agenda.

Typically, this person is from Finance or SalesOps department closely cooperating with Sales Managers, HR, and legal team.

Sales Manager

A person who is responsible for the actual creation of individual Compensation Agreements. The person enters conditions into agreements, negotiates details with Sales Agents, and is able to watch progress towards set goals. The Sales Manager also generates reports for those who are not able to access the system on their own. It could be an alternative name for Team Lead.

Sales Agent

A person who actually sells and is also the main recipient of compensations. It could be an alternative name for a Sales Representative, SalesRep, or Salesperson.

Scope of Usage

The Accelerator is a tool used in agendas of the above-mentioned roles. It helps them simplify, automatize, track or define compensation-related work tasks.

Creating Compensation Agreement

Compensation Agreement is one unit of the main Sales Compensation Plan. It can encapsulate one or more rules, conditions, exclusions, and time spans that are used for the calculation of the final sum for the payee. This definition can represent a legal contract.

Reporting Progress

Reporting is an activity that can be done on an agreement level or with a scope of several Sales Agents. Provided reports and dashboards are used for the analysis of the progress of the plan.

Plan Maintenance

Corrections of wrongly accounted transactions, new agreement revisions, definitions for a new period of a year, or replacements in the sales team are common activities associated with a well-defined plan.

Reporting and Dashboards

The ability to accurately track and compare progress towards defined goals is an important aspect of successful strategy execution. The package is distributed with a dashboard and set of reports supporting those processes.

Compensation Dashboard

Interactive reporting in dashboards is used for a deeper understanding of compensation agreements.

Users are restricted to see data of their subordinates and their own. You can read more on Dashboards Description (SC).

Overview Report

A document summarizing definitions of an agreement with results for each agreement line and total compensation value. This is the one-pager for the agreement in numbers.

YTD Report

It is a simple document showing agreement progress in the current calendar year. It goes into fewer details about each agreement line but focuses on results for the year.

Transaction List

This report is used for checking all transactions that are included in the calculation of the selected compensation agreement.

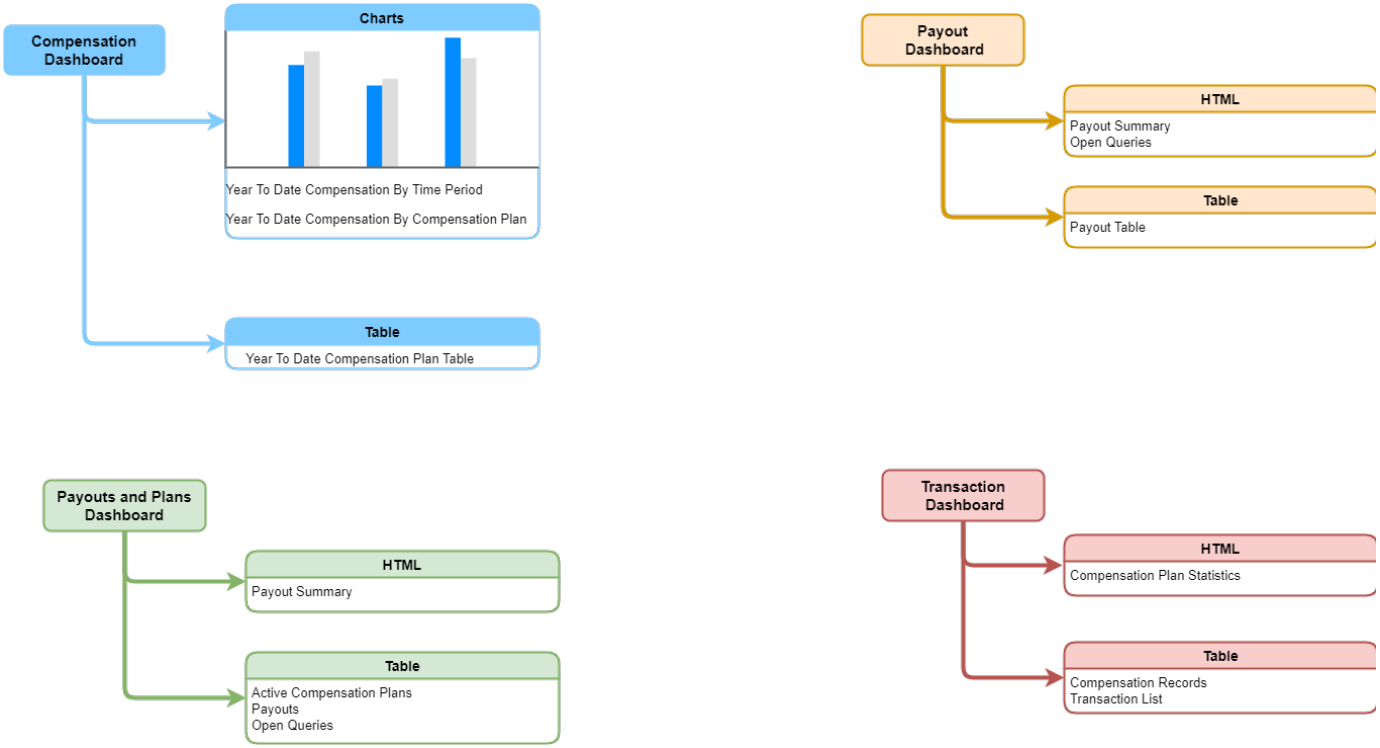
It can be used as an audit supporting tool or document generated when an agreement is fulfilled, see example SC_Transaction_List_Report_SAMPLE.pdf.

Dashboards Description (SC)

Sales Compensations (SC) has four dashboards used for overview, analysis and tracking of compensations.

- Compensation Dashboard – Provides a progress overview for individual sales agents for the current year.
- Payout Dashboard – Overview dashboard for tracking all payouts for SC Administrator.
- Payouts and Plans Dashboard – Personal dashboard for salespeople where they see planned and past payouts together with their plans.
- Transaction Dashboard – Analytical dashboard for SC Administrator that is used for troubleshooting down to transaction level.

> The dashboard access management is controlled by Sellers records from master data. If you see "There's no seller defined for user" message, it means your Pricefx user ID is not associated with any record.



Compensation Dashboard

This dashboard provides a progress overview for a particular sales agent for the current year. A central focus is to show income in form of a table and bar charts for Compensation Plans or time periods (months, quarters, ...).

Its primary use is for Sales Agents but it will give valuable insights to sales managers or SC Administrators as a supportive analytical tool.

Portlets

- **Year To Date Compensation** – A bar chart aggregating all compensations for the selected Sales Agent in time periods. If no specific Compensation Plan was selected, one bar combines all Compensation Plans together giving a total compensation sum that was paid out to the Sales Agent.
  Be careful, 'Time Period' input provides an aggregation period, e. g. Quarterly would show increments only in 2 bars if the compensation had a semi-annual payment period.
- **Year To Date Compensation By Compensation Plan** – A bar chart showing the total sum of all compensations in one agreement as a bar on the X-axis. The graph is more impactful for visualization of differences between Compensation Plans.
- **Year To Date Compensation Plan Table** – A summary table showing the breakup of Compensation Plans through Line Items (conditions) down to Compensation Record in a single table. A user is able to see which lines or periods are outliers.

User Inputs

Input Filter Name	Values	Notes
Seller	One Sales Agent	• Mandatory field • 'Compensation Plan' input is affected by the selected value.
Compensation Plan	Empty or one specific Compensation Plan	• Empty value compares all agreements that are assigned to the agent from the Seller input.
Time Period	Monthly, Quarterly, Semi-Annually, Annually	• Sets an aggregation period for the 'Year To Date Compensation' portlet.

Admin Dashboard

This dashboard is designed for administrators who can see all payouts for the seller(s) and time period they selected.

> Access to this dashboard is restricted to users with user group SC_Administrator (label: [SC] Administrator).

Portlets

- **Payout Summary** – Shows a total sum of all payouts from the filtered context. The second number is the total revenue generated.
- **Payout Table** – Provides a high level overview with a split between individual and group compensations. It has information on when the respective line was or will be paid.
  Numbers with a yellow background are incomplete summaries for the next month. Until their end date is finalized, their value can change.

User Inputs

Input Filter Name	Values	Notes

Seller(s)	Seller Group picker	• Optional • Empty value means the whole salesforce is selected.
Payout Year	Year drop-down picker	• Mandatory field • Shows only years where there were payouts for the selected Seller(s).
Payout Date	Date drop-down picker	• Mandatory field • Shows only dates of a selected Payout Year where there were payouts for the selected Seller(s).

Payouts and Plans Dashboard

This dashboard is a dedicated overview of Sales Agents' Compensation Plans and Payouts. The dashboard provides a list of Compensation Plans and their structure without having to access the Sales Compensations module. Additionally, users can also see their payouts (made or planned).

The dashboard is intended to be used by Sales Agents, Sales Managers as well as Sales Compensations administrators. Sales Agents can see only their own data, Sales Managers can see details for all their reports (based on the "Reports To" field in the Seller Master table) and SC Admins have unlimited selection.

Portlets

- **Payout Summary** - Quick summary of all payouts that have been paid in the selected period. Aside from the payout sum, there is also revenue so the user can see this KPI on the side of his commission.
- **Active Compensation Plans** - A table showing a list of all open plans. Clicking a Compensation Plan opens a detail. In the detail view, the user can see a simplified version of the configuration from the Compensation Plan stored in the Sales Compensations module. The view does not provide current progress but shows the definition on one screen when a line is clicked.
- **Payouts** - This table gives an overview of what has been paid and what is estimated for the next scheduled payment.
  Each Compensation Plan can have several payouts because they are linked to the conditions (items) of the plan. The user is able to see how much and when payouts occurred in past.
  Numbers with a yellow background are incomplete for the upcoming months. Until their due date is reached their value can change.

User Inputs

Input Filter Name	Values	Notes
Seller(s)	Seller Group picker	• Optional • Pre-selected and read only for Sales Agents. • Only reporting agents can be selected by Sales Managers. • Empty value means the whole salesforce is selected (based on permissions).

Payout Year	Year drop-down picker	• Mandatory field • Shows only years where there were payouts for the selected Seller(s).
Payout Date	Date drop-down picker	• Mandatory field • Shows only dates of a selected Payout Year where there were payouts for the selected Seller(s).

Transaction Dashboard

This dashboard is designed to provide an overview of all transactions that are included in the calculation of the selected Compensation Plan with the possibility to further filter on specific compensation record or transaction date.

> Access to this dashboard is restricted to users with the user group SC_Administrator (label: [SC] Administrator).

Portlets

- **Compensation Plan Statistics** - Statistical overview of the selected scope from Compensation Plans in a form of several tiles. The following information is represented on tiles:
  - Number of Transactions
  - Number of Negative Baseline Transactions
  - Number of Sales Agents
  - Number of unique SKUs (Products IDs)
  - Number of unique Customers
  - Total Baseline
- **Compensation Records** - Provides information about attributes for compensation records. It can contain just one line if the analysis is focused on one specific condition from the Compensation Plan. The "Id" column is an active link leading directly to the full detail of the Compensation Record, similarly for "Compensation Plan Id" which leads to the full detail of the Compensation Plan. The SC Administrator can follow the link and see a context that was not available on the dashboard. An example is "Calculation Baseline" showing raw transaction data for the selected compensation record.
- **Transaction List** - A list of selected columns from source transactions that were used as calculation baseline for the selected scope.

> Tip: Use Group By option on a column you would like to aggregate.

User Inputs

Input Filter Name	Values	Notes

Seller(s)	Seller Group picker	• Optional. • Pre-selected and read only for Sales Agents. • Only reporting agents can be selected by Sales Managers. • Empty value means the whole salesforce is selected (based on permissions).
Compensation Plan Type	Compensation types	• Optional. • Multi select. • Empty value means all Compensation Plan Types.
Compensation Plan	List of all Compensation Plans for a sales persons of the selected compensation type	• Optional. • Multi select. • Empty value means all Compensation Plans.
Compensation Record(s)	List of all Compensation Records associated with a given Compensation Plan	• Optional. • Multi select. • Empty value means all compensation records.
Date From	Date	• Empty value means no date filtering.
Date To	Date	• Empty value means no date filtering.

Get Started (SC)

- Installation (SC)
- Configuration (SC)

Installation (SC)

This tutorial will guide you through the installation of the Sales Compensation Accelerator.

In this section:

- Pre-requisites
- Installation Steps
  - Select Partition for Deployment
- Post-installation Steps
  - Add Indexing

- Enable React UI
- Schedule Plan Calculation Task
- Schedule Record Calculation Task
- Assign Access to Users
- Define Approval Workflow Logic for Adjustments
- Configure SC_AdjustmentWizardTransactionFieldMapping PP Table
- Configure SC_HeaderSellerDetailsMapping PP Table

Pre-requisites

Before you start, ensure that you have:

- Access to a partition on the Pricefx server (9.0 or newer). You will need:
  - Server URL
  - Partition name
  - Username and password for a partition user with sufficient rights for using the Accelerator
  - License on the partition must cover the Analytics and Sales Compensations modules
- Access to Pricefx PlatformManager
  - Username and password for PlatformManager user
  - The user must have the following permissions for your partition (to which you plan to deploy the Accelerator):
    - Permission *Marketplace Templates - deploy*
- Transaction data in the Datamart structure with required fields containing the following data:

Requirement description	Deployment label
The field name of a Value Base field is used to calculate the base value. In most cases, it will be a revenue, margin or gross margin field.	Baseline Value
The filed name of the Margin field.	Margin
The field name contains an identification of a sales agent.	Seller Id
The field name contains a customer ID in the Datamart.	Customer Id
The field name contains a product ID in the Datamart.	Product Id
The field name contains a product name. It is used for the *Transaction List* report only.	Product Name
The field name contains a pricing date in the Datamart.	Pricing Date
The field contains the main Datamart currency.	Currency

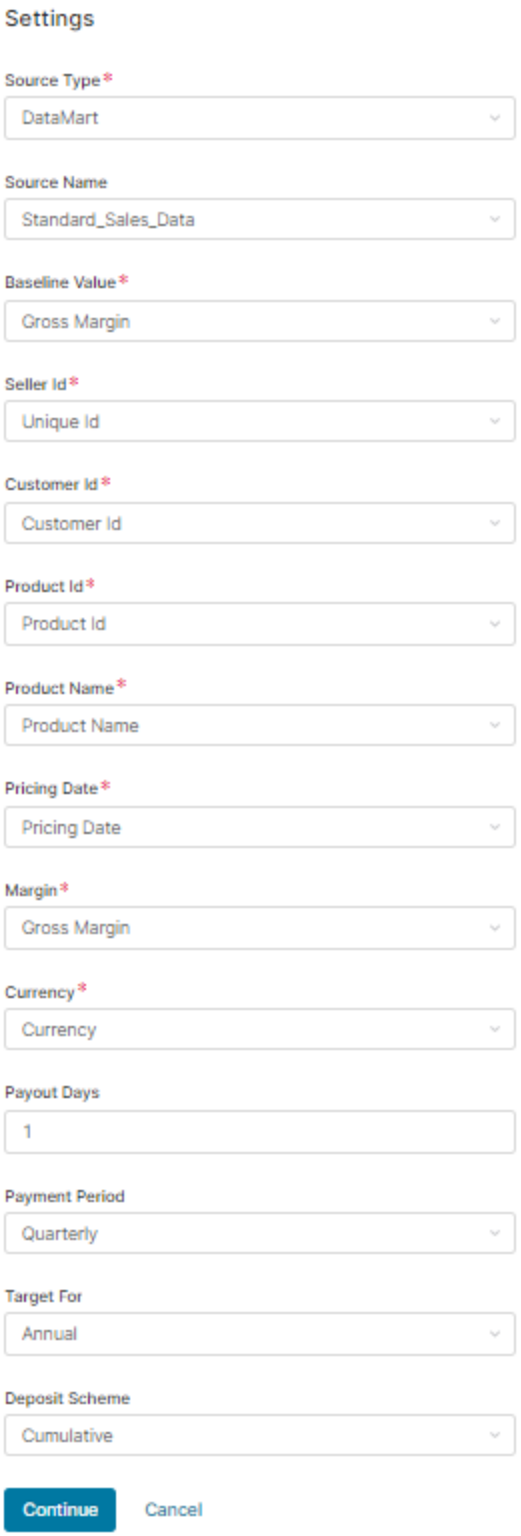
The package contains several components and their technical description is provided on the Components (SC) page.

Installation Steps

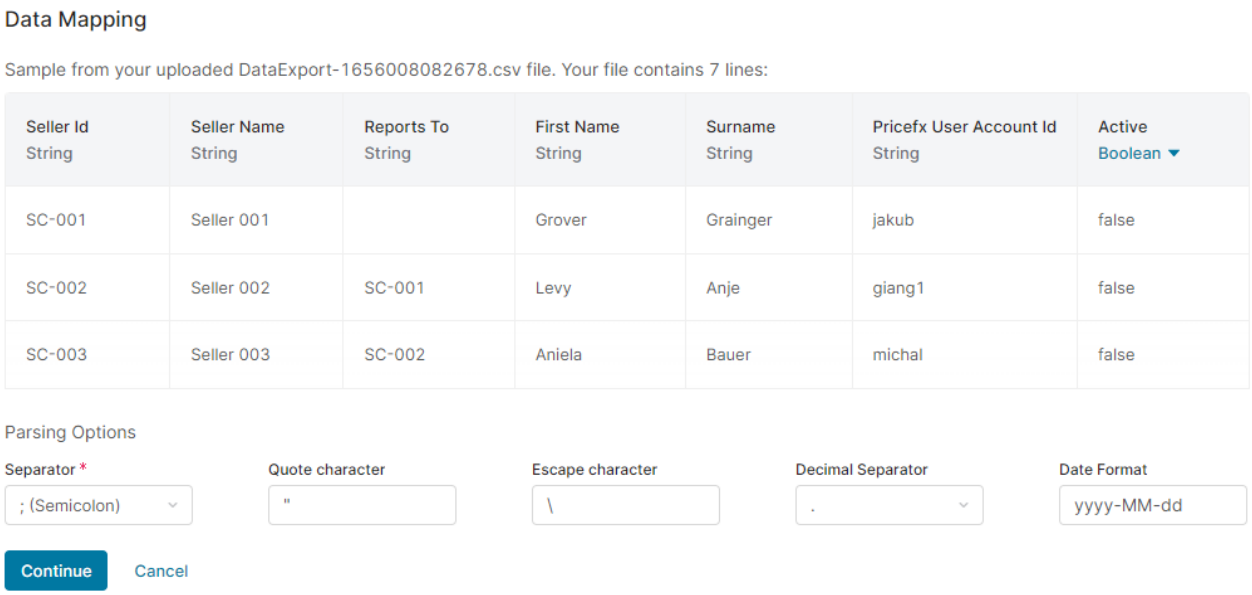
Select Partition for Deployment

1. In PlatformManager, navigate to **Marketplace > Accelerator Packages**, find the *Sales Compensation.*
2. Click **Deploy** and select a partition to which you want to deploy the package.
3. Click **Deploy**.

4. A warning dialogue will appear. After you read the warning text and you agree with the conditions, you can click **Continue**.
   If you need to leave the deployment process before it is finished, you can always come back later. The wizard will offer you to either start again, or continue in the previously started process.
5. Set up Datamart mapping of required fields from Datamart and default values for a few parameters.



6. Upload the seller master data and define the mapping.



7. The deployment is complete. Go to your partition and continue with the manual steps required after deployment.

Post-installation Steps

After the package is deployed to your partition and all automatic installation steps are done, you need to do a few manual tasks before you start with configuration and package adjustments to your specific business needs.

Add Indexing

To avoid performance issues when running Calculation Tasks, indexing for COARs on the partition should be added, for these fields: accrualType, attribute1, attribute20.

Example command:

```
MariaDB [pricefx_templates_dev]> create index typeAttr1Attr20 on
CompensationAccrualRecord(partition_id, accrualType, attribute1, attribute20)
using BTREE;
```

Enable React UI

The package is designed to work with the latest UI engine in the Sales Compensations module. For proper functionality is necessary to enable it manually.

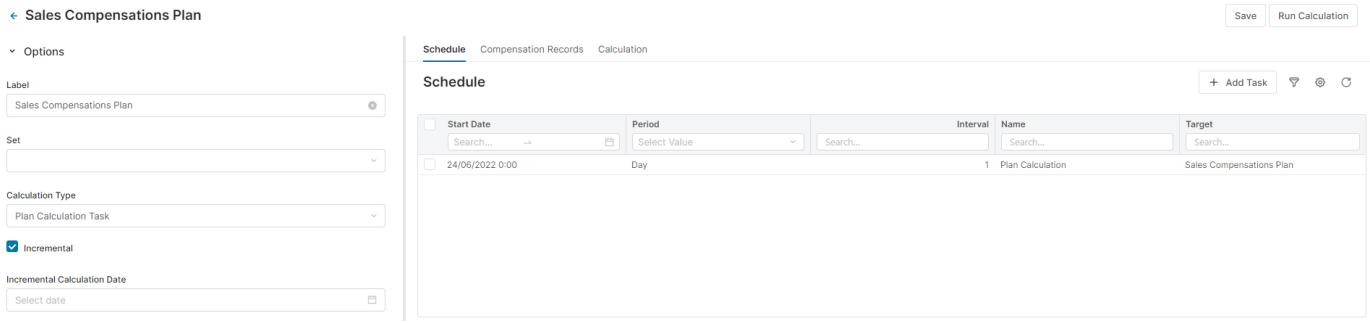
1. From the **Administration** menu go to the **Feature Flags** section.
2. Activate the following Feature Flags:
   `dashboard.useReactDashboard`
   `useReactFor.advancedFilter`
3. Log out and log back in for the changes to take effect.

Schedule Plan Calculation Task

In this step, you will create a task for the plan calculation scheduler.

1. Go to **Sales Compensations** > **Calculations**.
2. Add a new calculation.
3. Enter a label, e.g. 'SC_Calculation'.

4. Select a set 'Sales Compensations' and click the **Add** button.
5. Click the label of the newly added line.
6. Go to the Calculation tab.
7. As Logic/Formula choose '[SC] Sales Compensation'
8. As Feeder Formula choose '[SC] Compensation Record Calculation Feeder'.
9. Set the Calculation Type to Plan Calculation Task.
10. Optionally, enter StartDate and EndDate.
    Leave empty unless you need to calculate some specific time frame.
11. Click **Save** in the left upper corner.
12. Enable **Incremental** calculation on the left side of your screen.
13. Schedule the calculating task according to your data, e.g. daily.
14. In the Overview, you can schedule this job to run at a suitable interval.



Schedule Record Calculation Task

The steps for creating the Record calculation plan are almost the same as for Plan calculation, the only difference is in step 9 where the Calculation Type should be set to Record Calculation Task.

Assign Access to Users

The package comes with three predefined business roles which speed up configuration. You can adjust their default configuration and assign them to the users who have access to the system.

A detailed description of role definitions and their meaning is described in Product Info (SC) | Business roles.

1. From the **Administration** menu go to the **Access Admin > Business Roles Admin** section.
2. Check if all roles with the '[SC] ' prefix correspond with your expectations for minimal access and adjust them to your use cases.
3. Assign user accounts to their business roles from the Users tab of each role.

At this point, you completed the installation. Now, you should configure the package for specific business needs and get familiar with its use. The last thing to consider is the integration with a system that processes output compensations.

Define Approval Workflow Logic for Adjustments

For details how to set up the Workflow Package see this page. For the ApprovalCondition Company Parameters set the Workflow Type to Adjustment.

Configure SC_AdjustmentWizardTransactionFieldMapping PP Table

The Transaction matrix that is visible in the Adjustment custom form is configurable in the SC_AdjustmentWizardTransactionFieldMapping PP table. This table should be adjusted after the deployment to match the user data source setup.

Values that should be mapped:

- Unique ID
- Product ID
- Customer ID
- Seller ID
- Pricing Date
- Invoice Price
- Quantity
- Gross Margin

The Value column will initially be filled with "-ADJUST_ME-". The user should place there a name of a column that represents the appropriate field in the Data Source that will be used as a source for transactions. For example: Unique ID | uniqueIdColumn, Invoice Price | invoicePriceColumn

Configure SC_HeaderSellerDetailsMapping PP Table

The additional information about the seller that is displayed on the header level of a Compensation Plan is configurable in the SC_HeaderSellerDetailsMapping PP table. This table should be adjusted after the deployment to match the user master data setup and additional needs.

The structure of the table is as follows:

- Field label – Label of the field that will show in the header.
- Seller field – Here the user should place a name of the column that represents the matching field in the Seller master data. For example: Seller ID | sellerId.

Configuration (SC)

This tutorial will guide you through the configuration steps. After you have completed the installation, you can start with the configuration of the Sales Compensation (SC) Accelerator. You can customize the default configuration, extend types or remove those that are not needed.

In this section:

- Global Package Configuration
  - Company Parameters
- Plan Header Types
  - How to Customize Your Types
- Condition Types
  - How to Customize Your Types
  - How to Add a New Type
- Define Seller Information in Compensation Header
  - How to Customize Your Seller Information Display
- Define Adjustment Wizard Transaction Field in Adjustment Form
  - How to Customize Your Adjustment Wizard Transaction Field Display
- Define Approval Workflows
  - Compensation Plan Approval Workflow
  - Compensation Record Approval Workflow
  - Adjustment Custom Form Workflow

Global Package Configuration

The global configuration data for the package are stored in an element of Advanced Configuration Options. Many of those parameters can be overridden on the Condition Type.

1. From the **Administration** menu go to the System **Configuration** section.
2. Switch to **Advanced Configuration Options**.
3. Search for the `SC_AdvancedConfiguration` element where you can adjust the parameters listed below.
4. Adjust configuration keys according to your specifications.

A list of all parameters, their descriptions, possible values, and other notes are in Components (SC) | Advanced Configuration Options.

Sample of '**SC_AdvancedConfiguration**':

```
{
  "sourceType": "DM",
  "sourceName": "SC_TransactionData",
  "pricingDateFieldName": "PricingDate",
  "customerIDFieldName": "CustomerId",
  "productIDFieldName": "ProductId",
  "productNameFieldName": "ProductName",
  "baselineFieldName": "InvoicePrice",
  "invoicePriceFieldName": "InvoicePrice",
  "quantityFieldName": "Quantity",
  "marginFieldName": "GrossMargin",
  "sellerIDFieldName": "SellerId",
  "currencyFieldName" : "Currency",
  "payoutDays": 2,
  "paymentPeriod": "Quarterly",
  "quotaFor": "Annual",
  "depositScheme": "Non-Cumulative"
}
```

Company Parameters

To be able to apply a Customer filter input and Product filter input in a compensation plan, you need to update the mapping fields between Product/Customer master and Datamart fields.

Based on the Datamart structure and Product/Customer master in your partition, you need to update or add more mapping in two Company Parameters:

- SC_CustomerFieldMapping
- SC_ProductFieldMapping

Plan Header Types

How to Customize Your Types

1. Go to **Sales Compensations** > **Compensation Types**.
2. Select the line you want to edit.
3. Click the Edit button to open the customization dialogue.
4. Adjust parameters according to your needs by replacing default values with custom logic.
5. Click Save Changes to save and confirm changes.

A list of all parameters, their descriptions, possible values, and other notes are in Components (SC).

Sample of the 'Plan Header Type' which is deployed with the package:

```
Name (uniqueName): Individual Compensation
Header Logic (headerFormulaName): Sales Compensation Header
Compensation Plan Workflow (workflowFormulaName): Sales Compensation
Plan
Compensation Record Workflow (scRecordWorkflowFormulaName):
Compensation Record
```

Condition Types

This chapter describes procedures for manipulation with LineItem elements which are primary blocks of calculation for compensations inside the individual plan.

How to Customize Your Types

1. Go to **Sales Compensations** > **Condition Types**.
2. Look for items with the 'Sales Compensation' in the 'Pricing Logic (formulaName)' column.
3. Select the line you want to edit.
4. Click the Edit button to open the customization dialogue.
5. Adjust parameters according to your needs by replacing default values with custom ones.
6. Click Save Changes to confirm changes.

How to Add a New Type

1. Go to **Sales Compensations** > **Condition Types**.
2. Click 'Add Condition Type' and a configuration dialogue will open.
3. Fill in fields you want to have specifically defined in the condition type.
   - ⚠️ Be aware, parameters can inherit default values from Advanced Configuration Options. Empty field means the parameter inherits a value.
   - "Condition Type Name" shows a list of Sales Compensation. Options compatible with this package are listed in Type Overview (SC) | Condition Types.

4. Click Add to save your new type.

A list of all parameters, their descriptions, inheritance, possible values, and other notes are in section Components (SC) | Condition Type Attributes.

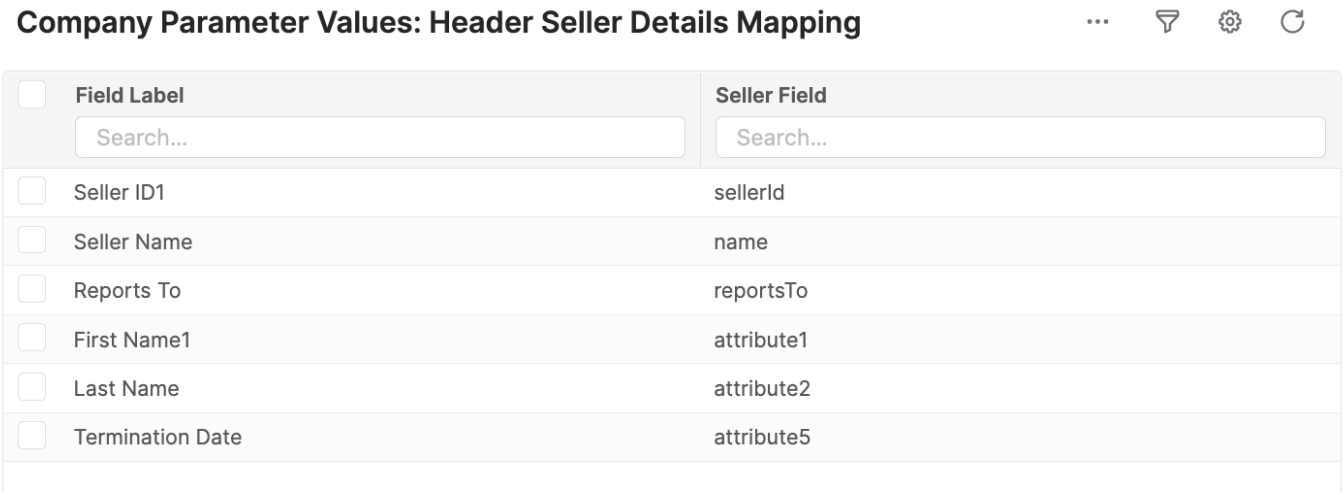
Define Seller Information in Compensation Header

Add a new read-only input which will show the detailed information about selected Seller input on the Compensation Plan Header.

How to Customize Your Seller Information Display

1. Go to Company Parameters.
2. Go to Pricefx Accelerators parent folder Sales Compensation folder.
3. Select SC_HeaderSellerDetailsMapping Company Parameter name.
4. Add a new record or adjust the existing one to display which the seller information you want.

Sample of SC_HeaderSellerDetailsMapping which is deployed with the package:



Define Adjustment Wizard Transaction Field in Adjustment Form

You can display any transaction field in the Adjustment Form (Adjustments under the Sales Compensation module).

How to Customize Your Adjustment Wizard Transaction Field Display

1. Go to Company Parameters.
2. Go to Pricefx Accelerators parent folder Sales Compensation folder.
3. Select SC_AdjustmentWizardTransactionFieldMapping Company Parameter name.
4. Add a new record or adjust the existing one to display which the Adjustment Field Name you want in the Adjustment Form.

Sample of SC_AdjustmentWizardTransactionFieldMapping which is deployed with the package:

Company Parameter Values: SC Adjustment Wizard Transaction Field Mapping

Adjustment Field Name	Datamart Field Name
Search...	Search...
Customer ID	-ADJUST_ME-
Invoice Price	-ADJUST_ME-
Margin	-ADJUST_ME-
Pricing Date	-ADJUST_ME-
Product ID	-ADJUST_ME-
Quantity	-ADJUST_ME-
Seller ID	-ADJUST_ME-

Define Approval Workflows

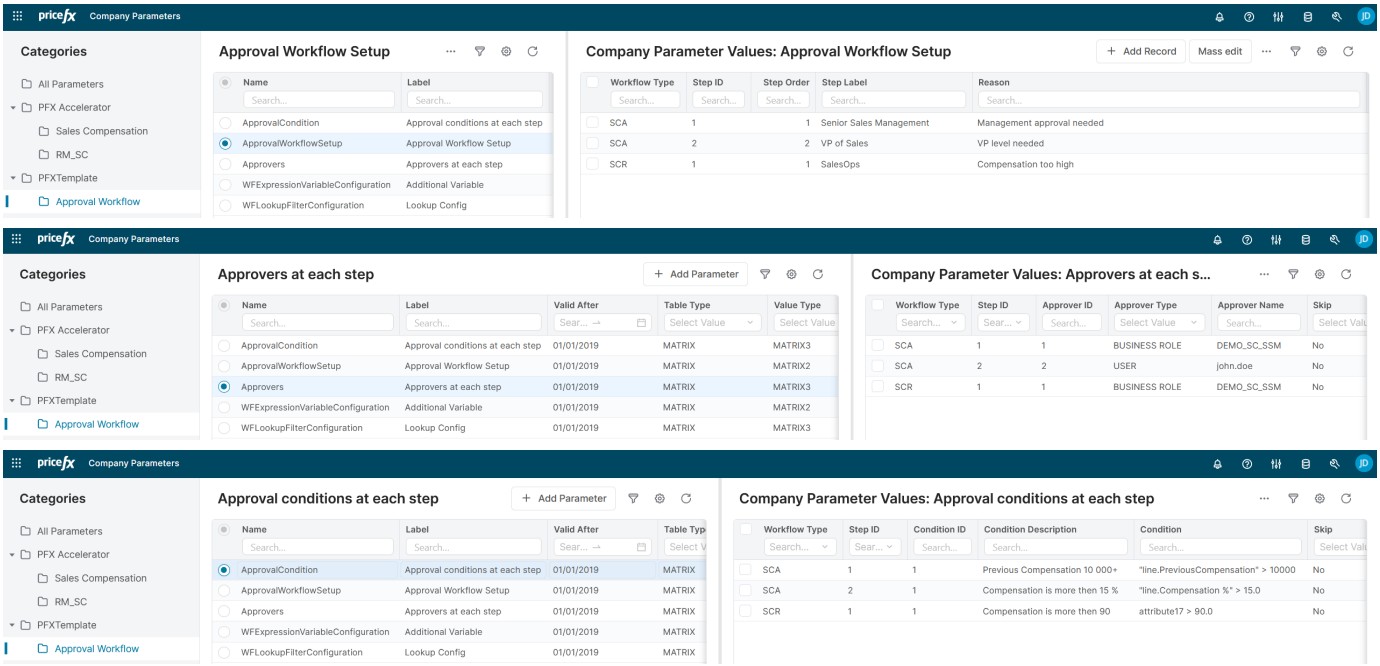
Two types of approval workflows can be configured to support business conditions and flow inside an organization. The installation package already deployed and configured workflow types (SCA and SCR, Adjustment) in **Administration** > **Logics** > **Workflow Logics** > **General Workflow Logics**. You just need to convert your business rules to workflow parameters in Company Parameters.

Compensation Plan Approval Workflow

In this step, you should configure workflow conditions for the Compensation Plan. You will follow detailed steps from the manual for the Accelerate Approval Workflow Package (AWP) which is part of this package.

1. Go to **Company Parameters** in the top menu.
2. Locate PFXTemplate > Approval Workflow.
3. Define workflow steps in the 'ApprovalWorkflowsSetup' parameter.
   - 'Workflow Type' must be *SCA*.
4. Define conditions in the 'ApprovalConditions' parameter.
   - The list of variables is on the Workflow Variables (SC) | Compensation Agreement approval workflows page.
5. Define approvals in the 'Approvers' parameter.
6. Test your workflow.
   You don't need to submit an plan to test it. Just recalculate and verify that the workflow is prepared to be triggered.

Example for a two-step workflow for Compensation Plan with conditions and approvals:



Compensation Record Approval Workflow

This workflow type configuration is identical to the one above. It uses **SCR** as the 'Workflow Type'.

An example from the previous step shows one step workflow configuration. The list of variables is on the Workflow Variables (SC) | Agreement Record approval workflows page.

To check the approvals for Compensation Records, you must approve the Compensation Plan first.

Adjustment Custom Form Workflow

This workflow type configuration is identical to the one above. It uses **Adjustment** as the 'Workflow Type'. An example from the previous step shows one step workflow configuration.

In the current development, you only need to define the approval condition as empty to display the approval step:

Reference (SC)

- Glossary (SC)
- Type Overview (SC)
- Components (SC)
- Workflow Variables (SC)
- Upgrade (SC)
- Troubleshooting (SC)
- Known Limitations (SC)

Glossary (SC)

This page lists specific terms, vocabulary, and definitions that are used in the context of the Sales Compensation package.

Term	Description
Compensation	A term used in the system as a final reward. In reality, it could be for some companies just a bonus, for other commission and in some cases incentives.
Incentive	An amount of money or non-monetary reward to motivate someone to achieve something.
Commission	Always in cash form, commission is an income payment. It could be a percentage of a product or service sold.
Sales Compensation Plan	It outlines sales agent' base salary as well as the company's commission and incentive program with strategy. Commission, bonus and incentive structure incentivize sales force to reach their objectives in order to earn a deserved reward.
Pay mix	The ratio of base salary to target compensations that make up On-Target Earnings (OTE).
On-Target Earnings (OTE)	Refers to an employee's pay mix made of basic salary and the additional variable component such as commission as their compensation.
SPIF	
Catch Up	True Up
Tollgate	Gate, Treshold
Clawback	
Attainment	
Cup	
Draw	
Quota	

See also the general glossary.

Type Overview (SC)

This section provides an overview and calculation examples for types that are distributed in the package.

- Agreement Header Types

- Condition Types
- Calculation Examples
  - [SC] Single Target Amount
  - [SC] Single Target Percent
  - [SC] Growth Absolute Amount
  - [SC] Growth Absolute Percent
  - [SC] Growth Percent Percent
  - [SC] Growth Percent Amount
  - [SC] Stepped Amount
  - [SC] Stepped Percent
- Agreement Records Calculation
  - Payout Value Calculation

Agreement Header Types

List of available types with information about input and outputs.

Name	Header Logic	Input	Payout	Description
[SC] Sales Compensation	[SC] Sales Compensation Agreement	Seller (single person)	Seller (single person)	Compensation is tracked for one Sales Agent who receives all payout money.

Condition Types

List of available types with their default setup. Below you will find how to modify the setup.

Type	Name	Classification	Target Type	Target Unit	Compensation Unit
Conditional	[SC] Single Target Amount	Maintain	Single	$	$
	[SC] Single Target Percent	Maintain	Single	$	%
Growth	[SC] Growth Absolute Amount	Growth	Multi	$	$
	[SC] Growth Absolute Percent	Growth	Multi	$	%
	[SC] Growth Percent Amount	Growth	Multi	%	$
	[SC] Growth Percent Percent	Growth	Multi	%	%
Stepped	[SC] Stepped Amount	Maintain	Multi	$	$
	[SC] Stepped Percent	Maintain	Multi	$	%

The dollar $ sign represents a currency in general, not an actual USD.

Calculation Examples

[SC] Single Target Amount

Inputs in the agreement:

- Target - $
- Compensation - $

Success formula: COMPENSATION_INPUT

Example

Target [input]	Compensation [input]	Base Line	Compensation [output]
$100,000	$1,000	$110,000	$1,000
$100,000	$1,000	$90,000	$0

[SC] Single Target Percent

Inputs in the agreement:

- Target - $
- Compensation Value - %

Success formula: (COMPENSATION_INPUT / 100) x BASELINE

Example

Target [input]	Compensation Value [input]	Base Line	Compensation [output]
$100,000	1%	$110,000	$1,100
$100,000	1%	$90,000	$0

[SC] Growth Absolute Amount

Comparison to some previous period - Month, Quarter, Year, Custom.

Inputs in the agreement:

- Target(s) - $
- Compensation Value(s) - $

Success formula: COMPENSATION_INPUT_FOR_TIER

Example

Definition:

Target [input] - Growth Tier	Compensation Value [input]
$10,000	$100

$25,000	$300
$100,000	$10,000

Result:

Growth Base Line (Current – Previous)	Compensation [output]
$5,000	$0
$30,000	$300
$150,000	$10,000

[SC] Growth Absolute Percent

Comparison to some previous period – Month, Quarter, Year, Custom.

Inputs in the agreement:

- Target(s) – $
- Compensation Value(s) – %

Success formula: (COMPENSATION_INPUT_FOR_TIER / 100) x BASELINE

Example

Definition:

Target [input] – Growth Tier	Compensation [input]
$10,000	1%
$25,000	2%
$100,000	5%

Result:

Base Line	Compensation [output]
$5,000	$0
$25,000	$500
$150,000	$7,500

[SC] Growth Percent Percent

Comparison to some previous period – Month, Quarter, Year, Custom.

Inputs in the agreement:

- Target(s) – %
- Compensation Value(s) – %

Success formula: (COMPENSATION_INPUT_FOR_TIER / 100) x BASELINE

Example

Definition:

Target [input] – Growth Tier %	Compensation Value [input]
2%	1%
5%	3%
10%	5%

Result:

Base Line	Calculation Base Line	Compensation [output]
1%	$10,000	$0
2%	$100,000	$1,000
11%	$100,000	$5,000

[SC] Growth Percent Amount

Comparison with some previous period – Month, Quarter, Year, Custom.

Inputs in the agreement:

- Target(s) – %
- Compensation Value(s) – $

Success formula: COMPENSATION_INPUT_FOR_TIER

Example

Definition:

Target [input] – Growth Tier %	Compensation Value [input]
2%	$1,000
5%	$10,000
10%	$25,000

Result:

Base Line	Compensation [output]
1%	$0
2%	$1,000
11%	$25,000

[SC] Stepped Amount

Inputs in the agreement:

- Target(s) – $
- Compensation Value(s) – $

Success formula for step 1: COMPENSATION_INPUT_FOR_STEP1

Success formula for step 2: COMPENSATION_INPUT_FOR_STEP1 + COMPENSATION_INPUT_FOR_STEP2

Example

Definition:

Target [input] – Step	Compensation Value [input]
$10,000	$100
$50,000	$500
$100,000	$5,000

Result:

Base Line	Compensation [output]
$5,000	$0
$15,000	$100
$110,000	*$100 + $500 + $5,000* = $5,600

[SC] Stepped Percent

Inputs in the agreement:

- Target(s) – $
- Compensation Value(s) – %

Success formula for step 1: (COMPENSATION_INPUT_FOR_STEP1 / 100) x (BASELINE - TARGET_STEP1)

Success formula for step 2: ((COMPENSATION_INPUT_FOR_STEP1 / 100) x TARGET_STEP1) + ((COMPENSATION_INPUT_FOR_STEP2 / 100) x (BASELINE - TARGET_STEP2))

Example

Definition:

Target [input] – Step	Compensation Value [input]
$10000	1%
$50000	3%

$100000	10%

Result:

Base Line	Compensation [output]
$5000	$0
$15000	*($5,000 * 1%)* = $50
$110000	*($40,000 * 1%) + ($50,000 * 3%) + ($10,000 * 10%) = $400 + $1,500 + $1,000* = $2,900

Agreement Records Calculation

Payout Value Calculation

The "Annual" target needs a Deposit Scheme. The scheme decides if the calculation is split by the number of periods in the target time span.

Input information:

- Condition Type: [SC] Single Target Amount
- Payment Period: Quarterly
- Target For: **Annual**

Target input:

- Target Input: 100,000.0
- Compensation Input: 1,000.0

As the payment period is each quarter, we have 4 agreement records with the following data:

Agreement (Rebate) Record	Period	Period Sale	Cumulative Sale
RR01	Q1	90,000	90,000
RR02	Q2	11,000	101,000
RR03	Q3	49,000	150,000
RR04	Q4	50,000	200,000

Non-Cumulative Example

The calculation is driven by following an execution pattern.

```
If ( Cumulative_Sale > Target_Input ) then
    Compensation_Value = Compensation_Input / 4
end
```

Note: Number 4 stands for a number of agreement records (Rebate Records without internalization applied).

Agreement (Rebate) Record	Period	Compensation Value
RR01	Q1	0
RR02	Q2	250
RR03	Q3	250
RR04	Q4	250

Cumulative Example

The calculation is driven by following an execution pattern.

```
If ( Cumulative Sale > Target_Input ) then
    Compensation_Value = ( Compensation_Input / 4 ) *
Current_Period_Index - Accrual
end
```

Notes:

- Number 4 stands for a number of agreement records (Rebate Records without application text changes).
- Current Period Index starts with 1.
- Accrual is the cumulative compensation.

Agreement Record	Period	Current Period Index	Accrual	Compensation Value
RR01	Q1	1	0	0
RR02	Q2	2	0	500
RR03	Q3	3	500	250
RR04	Q4	4	750	250

Components (SC)

On this page, you will find a technical description and a list of components used in the package.

- Groovy Overview
  - Logics
  - Libraries
- Advanced Configuration Options
- Company Parameters

    - RM_SC_ConditionTypes
    - SC_CustomerFieldMapping
    - SC_ProductFieldMapping
    - Dependency Parameters
- Attributes
    - Agreement Header Type Attributes
    - Condition Type Attributes
    - Agreement Record Attributes
    - Sellers Attributes
- Publishing Templates
- Dashboards
- Access Management
- Other Components
    - Approval Workflows

Groovy Overview

Logics

Logic Name	Default Label	Description
SC_Compensation	Sales Compensation	This logic processes all compensation types. If you need to add more input/output elements for the compensation, add them in this logic.
SC_CompensationHeader	Sales Compensation Header	Header logic for a personal compensations.
SC_CompensationRecordCalculationFeeder	Compensation Record Calculation Feeder	Finds and calculates agreement records.
SC_Dashboard_Compensations		Input generation, portlet generation, data fetching, etc.
SC_Dashboard_Compensations_Configurator	Compensation Dashboard Configurator	Builds a Seller input for dashboards.
SC_OverviewReport	Overview Report	Logic supporting report of the same name.
SC_SalesCompensationAgreementReport	Sales Compensation Agreement Report	Logic supporting report of the same name.
SC_TransactionsReport	Transactions Report	Logic supporting report of the same name.
SC_YTDReport	Year To Date Report	Logic supporting report of the same name.

Libraries

Library Name	Default Label	Description
SC_Compensation ProcessingLib	Sales Compensation Processing Library	Provides common functions used in the compensation package such as utilities for dates manipulation, inputs, record fetching, etc.
SC_Compensation TypesLib	Condition Type Library	Defines all compensation types. When you have a new condition type, add a new element to this library to define functions for this condition type.
SC_Compensation DashboardsLib	Sales Compensation Dashboard Library	Provides common functions used in dashboards of the compensation package.
SharedLib		Provides common functions used in the compensation package. Provided by Shared Groovy Library dependency.
ApprovalWorkflow		Provides function for approval workflow logics. Provided by Approval Workflow Library dependency.
FormulaEvaluator		Used in Approval Workflow. Provided by Formula Evaluator Library dependency.
HighchartLibrary		Provides "Highcharts" functions used in dashboards. Provided by Dashboards Accelerator Library dependency.

Advanced Configuration Options

Package configuration is done from this central point in the system. Option values are configured mostly during a package deployment process and can be adjusted later. Some options can be overridden on a Condition Type or Line Item level where the lower level has a priority.
The following table shows a list of all configuration parameters stored in `SC_AcceleratorConfiguration`.

Name	Description	Condition Type overwritable
datamartName	Name of Datamart used to query and allocate the compensation value	
datamartBaselineFieldName	The field name of a Value Base field is used to calculate the base value. In most cases, it will be a revenue or margin field.	
datamartSellerIDFieldName	The field name contains an identification of a sales agent.	
datamartCustomerIDFieldName	The field name contains a customer ID in the Datamart.	

datamartProductIDFieldName	The field name contains a product ID in the Datamart.	
datamartProductNameFieldName	The field name contains a product name. It is used for the *Transaction List* report only.	
datamartPricingDateFieldName	The field name contains a pricing date in the Datamart.	
datamartCurrencyFieldName	The field contains the main Datamart currency.	
payoutDays	A number of days after the end date of the payment period when the payout happens. The parameter defines the default "Payout Days" for Condition Types that do not have it filled in directly.	✓
targetFor	The target defined on the line item is evaluated for each payment period (see *paymentPeriod*) or annually. The parameter defines the default "Target For" for Condition Types that do not have it filled in directly. Select one of these options: • **Payment Period** – Gets a base value for every period, then compares and calculates the compensation. • **Annual** – Gets a base value for the whole year, then compares and calculates the compensation.	✓
paymentPeriod	A parameter setting a frequency of payouts. The parameter defines the default "Payment Period" for Condition Types that do not have it filled in directly. The user can specify it also on a Line Item level. Values: "Monthly", "Quarterly", "Semi-Annually", "Annually"	✓
depositScheme	A parameter providing guidance on compensation accumulation across payment periods. If "targetFor" is "Annual", you need to define how to calculate the compensation value: • **Non-Cumulative** – The compensation value is calculated for the current period based on a cumulation of the base value, then divided by the number of periods (12 months or 4 quarters, 2 for semi-annually, 1 year). • **Cumulative** – The same as with calculation of Non-Cumulative above, but the compensation value of the previous period is excluded.	✓

Company Parameters

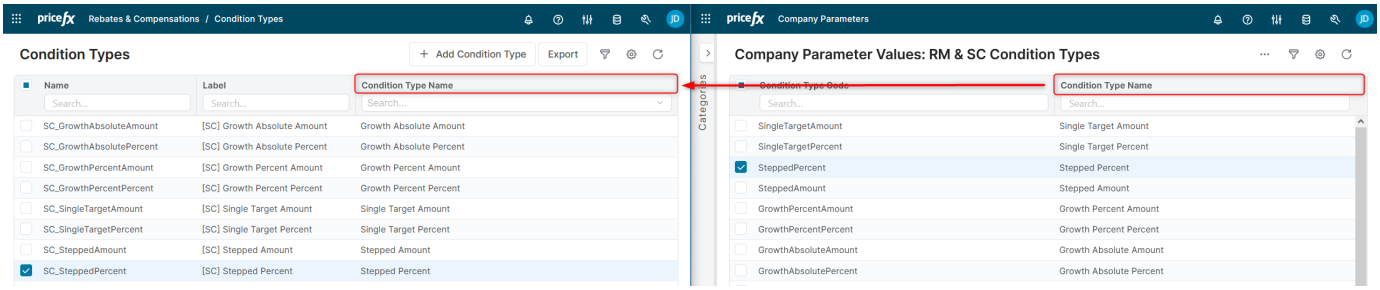
The following supporting configuration parameters are deployed with the package and are used for configuration.

RM_SC_ConditionTypes

This parameter saves the condition type code and condition type name. If the package supports a new condition type, you need to add a new row in this table.

The value of the "Condition Type Name" column will be shown in the "Condition Type Name" (attribute4) column from the Condition Types table from the Rebates module.

Column Name	Description
Condition Type Code	Must match the element name in Groovy Library "SC_CompensationTypesLib" (and "RebateTypesLib" from Rebate Manager if deployed together).
Condition Type Name	Will be shown in "Condition Type Name".



SC_CustomerFieldMapping

This parameter maps the **Customers** master table attributes (source fields) with the datamart field names. You should check it and update based on the master table structure in your partition as explained in the configuration documentation.

Column Name	Column Label	Description
name	Customer Master Field Name	Attribute name (source field) in Customers
attribute1	Datamart Field Name	Map with a field name in Datamart

SC_ProductFieldMapping

This parameter maps the **Products** master table attributes (source fields) with the Datamart field names. You should check it and update based on the master table structure in your partition as explained in the configuration documentation.

Column Name	Column Label	Description
name	Product Master Field Name	Attribute name (source field) in Products

attribute1	Datamart Field Name	Map with a Field Name in Datamart

Dependency Parameters

List of parameters that come from dependencies and have an effect on the package behavior.

- CurrencySymbols
- ApprovalCondition
- ApprovalWorkflowSetup
- Approvers
- WFExpressionVariableConfiguration
- WFLookupFilterConfiguration

Attributes

This section enlists attributes for various elements used for the package functionality.

Agreement Header Type Attributes

No extra attributes are used for Agreement Header Types.

Condition Type Attributes

Condition type attributes are needed for the sales compensation logic to calculate the compensation value and agreement records.

The package deploys also attributes used for functions in the Rebate Manager Package. Those are shown in light grey color.

Attribute	Attribute Name	Default Label	Type	Description	Advanced Configuration override
formulaName	formulaName	Pricing logic	Drop-down list	Sets a logic to run. The value should be "Compensation Logic".	
attribute1	CustomerSelectionLevel	Customer Selection Level	Drop-down list		
attribute2	CustomerFilterLogic	Customer Filter Logic	Text field		
attribute3	PaymentPeriod	Payment Period	Drop-down list	Defines the payment period for a condition type. If users do not set a value for this attribute, there is the "Payment Period" input in the compensation agreement with the same values. Values: "Monthly", "Quarterly", "Semi-Annually", "Annually"	✓
attribute4	ConditionTypeName	Condition Type Name	Drop-down list	Sets a condition type template for the current condition type. The value is set to "Condition Type Name" in Company Parameters.	

attribute5	PayoutDays	Payout Days	Number	Enter the number of days after the end date of the payment period when the user wants the payout.	✓
attribute6	BaseSource	Rebate Source Name	Text field		
attribute7	BaseFieldValue	Rebate Base Field Value	Text field		
attribute8	BaseFieldDate	Rebate Base Field Date	Text field		
attribute9	TargetFor	Target For	Drop-down list	Defines the target input for a period or annually. Values: "Payment Period", "Annual"	✓
attribute10	SourceType	Source Type	Drop-down list		
attribute11	DepositScheme	Deposit Scheme	Drop-down list	Defines the calculation type for a compensation value. If Target For = "Payment Period", users do not need to set a value for this attribute. Values: "Cumulative", "Non-Cumulative"	✓
attribute12	BaseFieldCustomer	Rebate Base Field Customer ID	Text field		
attribute13	BaseFieldProduct	Rebate Base Filed Product ID	Text field		
attribute14	BaseFieldQuantity	Rebate Base Field Quantity	Text field		
attribute15	ProductFilterLogic	Product Filter Logic	Text field		

Agreement Record Attributes

The following extra information is provided.

ℹ The package deploys also attributes used for functions in the Rebate Manager Package. Those are shown in light grey color.

Attribute	Attribute Name	Default Label	Description	Accelerator
attribute1	CurrentBaselineValue	Current Baseline Value	Baseline value (revenue or margin ...) in this period	SC, RM
attribute2	CurrentRebate	Current Rebate		RM
attribute3	PayToID	Pay To ID	The ID of a Sales Agent who will be paid	SC, RM

attribute4	CurrentBaselineQuantity	Current Baseline Quantity		RM
attribute5	ForecastBaselineValue	Forecast Baseline Value		RM
attribute6	ForecastQuantity	Forecast Quantity		RM
attribute7	Forecast	Forecast		RM
attribute8	AccrualForecastBaselineValue	Accrual Forecast Baseline Value		RM
attribute9	AccrualForecast	Accrual Forecast		RM
attribute10	TruesUp	Trues Up		RM
attribute11	AccrualMethod	Accrual Method		RM
attribute12	SalesGoalIncreasePct	Sales Goal Increase %		RM
attribute13	ForecastType	Forecast Type		RM
attribute14	AccrualForecastQuantity	Accrual Forecast Quantity		RM
attribute15	GeneralFilter	General Filter	Filter which was used as Compensation Agreement input.	SC
attribute16	Currency	Currency	Compensation Currency	SC
attribute17	CurrentCompensation	Current Compensation	Amount of money which will be paid	SC

Sellers Attributes

The following extra information is expected to be in the Sellers table.

Attribute	Label	Type	Is Mandatory?
sellerId	Seller ID	String	✅
name	Seller Name	String	✅
attribute1	First Name	String	❌
attribute2	Surname	String	❌
attribute3	Reports To	String	✅
attribute4	Pricefx User Account Id	String	✅
attribute5	Active	String	❌

Publishing Templates

The following templates are provided.

Looks like the page 'Installation (SC)' does not contain Multiple Excerpts (excerpt with key) or the given key 'sc_components_publishing_templates' is wrong. If problem persists, you can contact support at support@bitwelt.atlassian.net.

Dashboards

Table of dashboards included in the package.

Name	Logic	Description
Compensation Dashboard	SC_Dashboard_Compensations	Shows a compensations value of each compensation type from the beginning of the year to the current date.

Access Management

A list of predefined business roles and user groups with minimal access rights.

Business Roles

Business Role	Assigned User Roles
SC_Administrator	DASHBOARDADMIN, I18NADMIN, PA_DATAANALYZER, PA_DATAMANAGER, PA_SCHEMAEDITOR, PB_CUSTOMERS_RO, PB_PARAMETERS, PB_PRODUCTS_RO, RM_REBATEAGREEMENTS_ADMIN, RM_REBATEMANAGER, RM_REBATERECORDS_ADMIN, RM_SELLERSMANAGER, WF_ADMIN, WF_BUILDER
SC_SalesManager	PB_CUSTOMERS_RO, PB_PRODUCTS_RO, RM_REBATEAGREEMENTS, RM_REBATERECORDS
SC_SalesAgent	PB_CUSTOMERS_RO, PB_PRODUCTS_RO, RM_REBATEAGREEMENTS_RO, RM_REBATERECORDS_RO

For more details see User Roles and https://pricefx.atlassian.net/wiki/spaces/ACCDEV/pages/3807805649/Product+Info+SC#Business-Roles.

User Groups

- SC_SalesManager

Other Components

Approval Workflows

The Approval Workflow Library is part of the package as a dependency. It is extended with new types for sales compensation context. For details see Accelerate Approval Workflow Package (AWP).

Workflow Type	Workflow Name	Logic Name
SCA	Compensation **Agreement** Approval Workflow	SC_SalesCompensationAgreement
SCR	Agreement **Record** Approval Workflow	SC_AgreementRecord

Workflow Variables (SC)

This page provides a list of variables used for building conditions for approval workflows inside the package. It provides support information for Configuration (SC).

- Compensation Agreement Approval Workflows
- Agreement Record Approval Workflows

Compensation Agreement Approval Workflows

SCA is the name of Workflow Type that must be used for workflows you build on top of any Compensation Agreement. SCA must be used in ApprovalWorkflowSetup, ApprovalCondition and Approvers tables inside Company Parameters.

A list of all possible variables is available also in Element Name in Accelerators from Approval Workflow Package.

Header - Label	Element Name	Example
Start Date	startDate	startDate == "2021-01-31"
End Date	endDate	endDate == "2021-01-31"
Payout Date	payoutDate	payoutDate == "2021-01-31"
Seller	seller	Seller == "SC-001"
Currency	currency	header.currency == "EUR"
Header - Compensation Detail		
Previous Compensation	previousCompensation	previousCompensation >= 10000
Current Compensation	currentCompensation	currentCompensation >= 50000
Line Item - Label		
Condition Type	rebateType	rebateType = "SC_SingleTrargetPercent"
Input		
Target	line.Target	"line.Target" > 100
Compensation	line.Compensation	"line.Compensation" == 2.0
Compensation %	line.Compensation %	"line.Compensation %" > 10
Payment Period	line.Payment Period	"line.Payment Period" == "Annually"
Line Item - Compensation Detail		

Previous Baseline Value	line.PreviousBaselineValue	"line.PreviousBaselineValue" > 127
Current Baseline Value	line.CurrentBaselineValue	"line.CurrentBaselineValue" > 1200
Previous Compensation	line.PreviousCompensation	"line.PreviousCompensation" > 1000
Current Compensation	line.CurrentCompensation	"line.CurrentCompensation" >= 1200
Line Item - Compensation Type Info		
Target For	line.TargetFor	"line.TargetFor" == "Annual"
Deposit Scheme	line.DepositScheme	"line.DepositScheme" == "Non-Cumulative"
Payment Period	line.PaymentPeriod	"line.PaymentPeriod" == "Monthly"
Seller Name	line.SellerName	"line.SellerName" == "John Doe"
General Filter	<not supported>	
Customer(s)	<not supported>	
Product(s)	<not supported>	

Agreement Record Approval Workflows

SCR is the name of Workflow Type that must be used for workflows you build on top of any Agreement Record. SCR must be used in ApprovalWorkflowSetup, ApprovalCondition and Approvers tables inside Company Parameters.

A list of all possible variables is available also in Element Name in Accelerators from Approval Workflow Package.

When the condition is set up, use the field name and do not add any prefixes; see the table below with examples having specific meaning for this package.

Label	Element Name	Example
Current Baseline Value	attribute1	attribute3 > 1000
Pay To Id	attribute3	attribute3 == "SC-001"
Currency	attribute16	attribute16 == "EUR"
Current Compensation	attribute17	attribute17 >= 1500

Upgrade (SC)

This tutorial will guide you through the upgrade of the Sales Compensation Accelerator.

In this section:

- Pre-requisites
- Upgrade Steps

Pre-requisites

Before you start, ensure that you have:

- Access to a partition on the Pricefx server (8.0 or newer). You will need:
  - Server URL
  - Partition name
  - Username and password for a partition user with sufficient rights for using the Accelerator
- Access to Pricefx PlatformManager
  - Username and password for PlatformManager user
  - The user must have the following permissions for your partition (to which you plan to deploy the Accelerator):
    - Permission *Marketplace Templates - deploy*
- Familiarize yourself with the steps required after the upgrade – Manual Upgrade Steps (SC).
- Optionally, read about changes in Release Notes (SC).

Upgrade Steps

Select Partition for Deployment and upgrade logics.

1. In PlatformManager, navigate to **Marketplace** > **Accelerator Packages**, find the *Sales Compensation - Upgrade.*
2. Click **Deploy** and select a partition to which you want to upgrade.
   Only logics are deployed. The configuration remains without changes.
3. Click **Deploy**.
4. A warning dialogue will appear. After you read the warning text and you agree with the conditions, you can click **Continue**.
5. The first part of the upgrade is complete. Go to your partition and continue with the manual steps required after the upgrade.
   The exact process depends on the original version of the package, you will find all details in Manual Upgrade Steps (SC).

Troubleshooting (SC)

In this section, you can find troubleshooting tips that will help you during the support procedure.

Where do I find a package version?

For efficient communication with the support team, you should know which package version is used.

1. Go to **Company Parameters**.
2. Search for the 'deployedAccelerators' parameter.

3. Find the key 'sales-compensation', its value contains 'templateVersion'.
4. Version is a value stored in 'templateVersion'.

Known Limitations (SC)

Adjustments Limits

In version 1.4.0, when you submit the Adjustment, with filters on Transactions data returning data exceeding 25 000 rows, you will often get timeout. This issue will be resolved once the support for mass addOrUpdate for COARs is added: PFUN-17298 - Support COAR type in mass api.addOrUpdate() operation IN PROGRESS .

Warnings in Compensation Plan

In version 1.4.0, warnings on the Compensation Plan level are not displayed properly. The issue can be tracked in this ticket: PM-2369 - (BE) api.addWarning doesn't work in Sales Compensation and Rebate Manager BLOCKED .

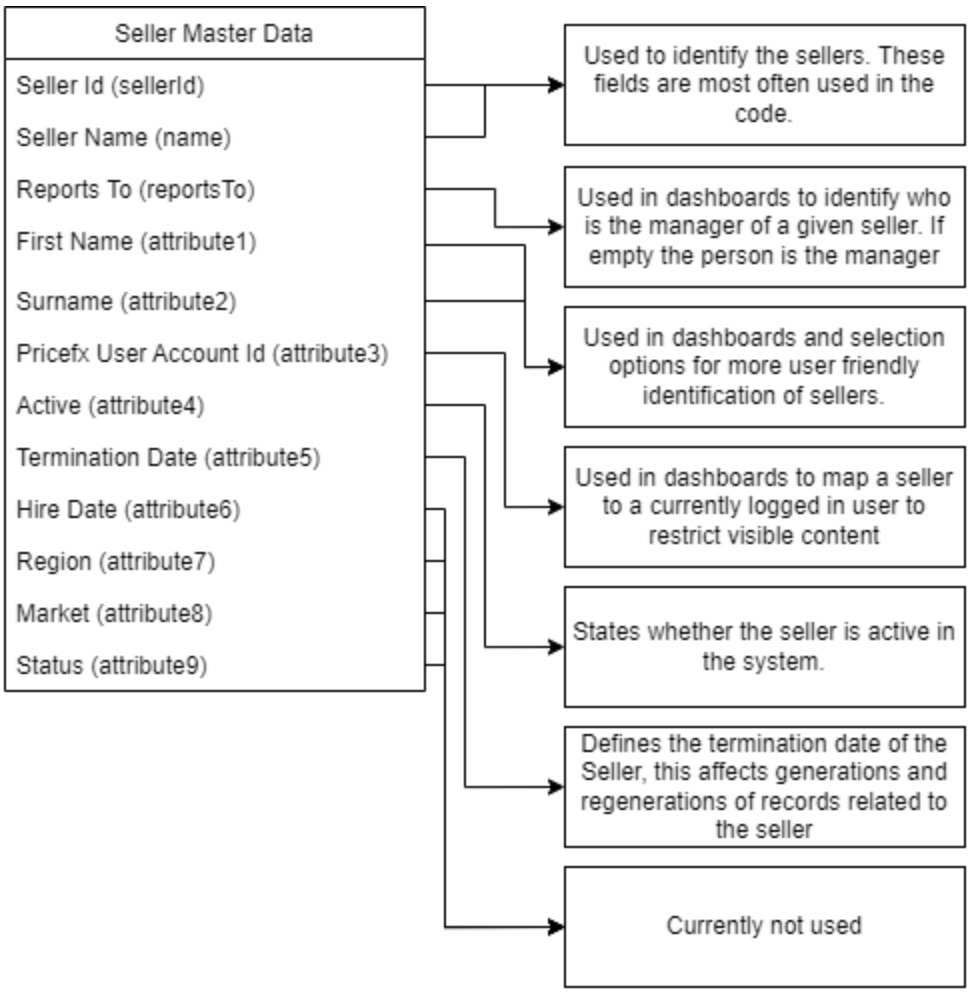
These warnings should be displayed if:

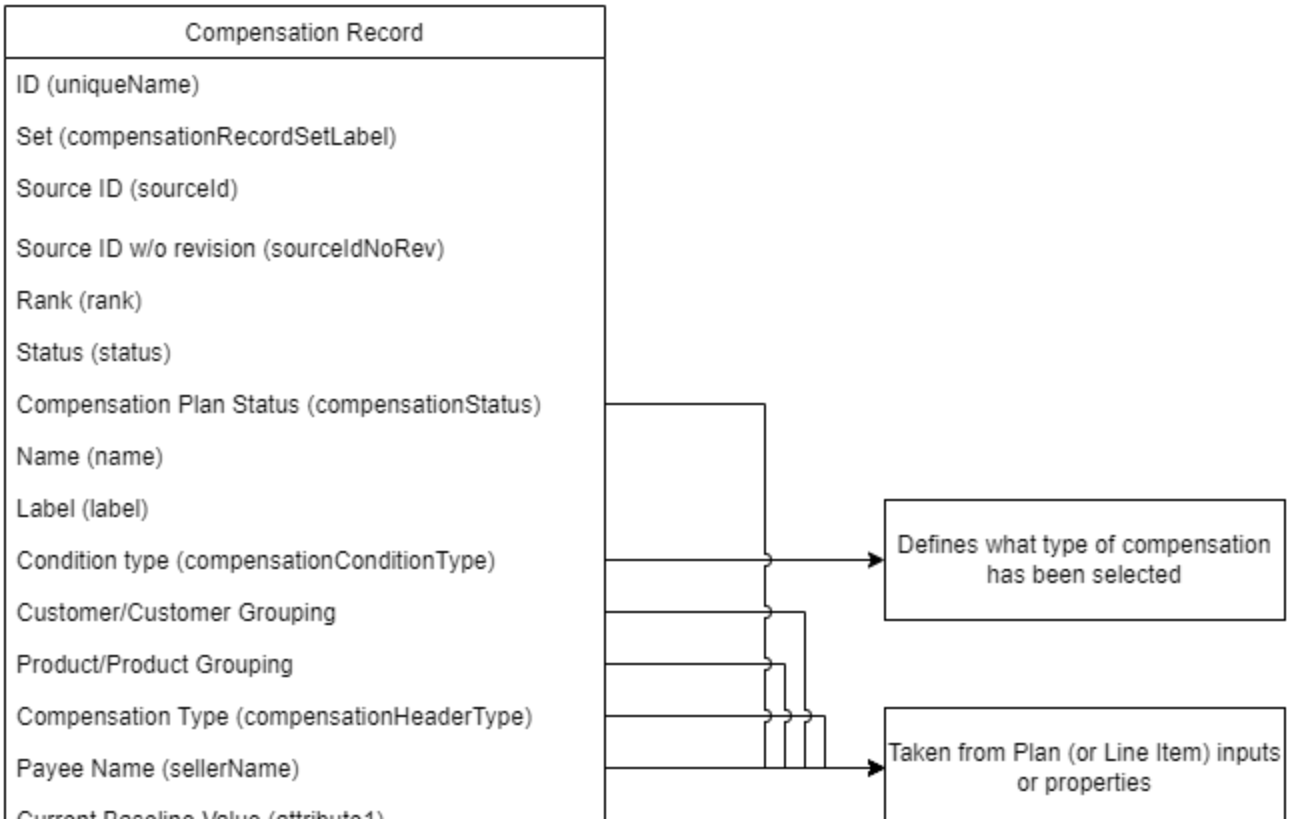
- Any of the sellers selected in the Seller input are terminated.
- Any of the sellers selected in the Seller input have a missing salary.
- Any of the sellers selected in the Team input have been skipped because of invalid validity dates.
- Any of the sellers selected in the Team input have been skipped because validity dates contradict planned start/end date settings.

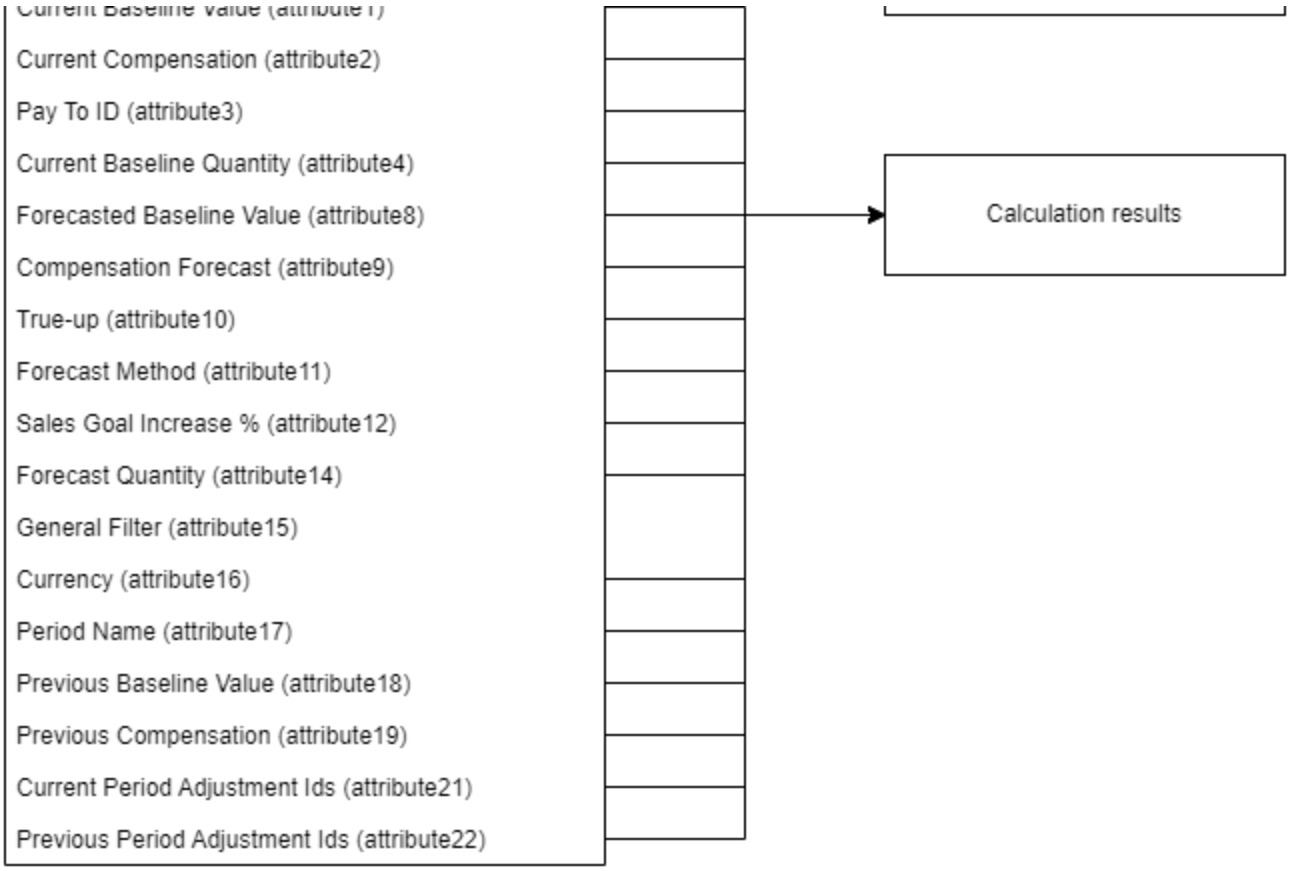
Technical Information (SC)

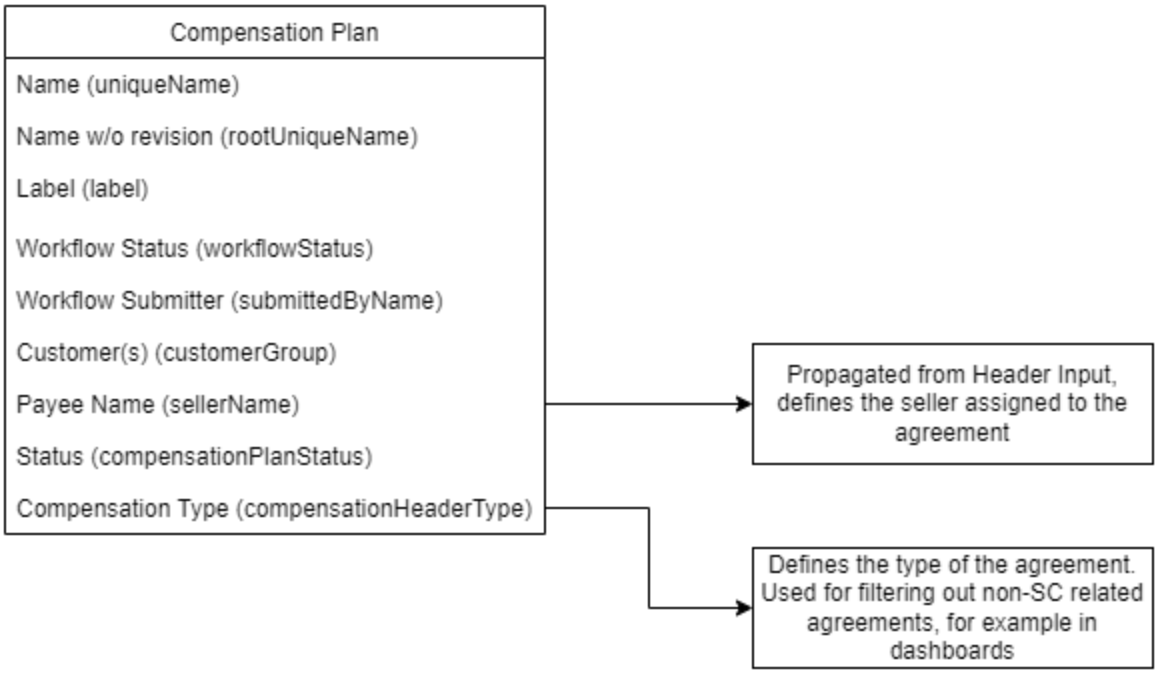
- Architecture Documentation (SC)
- Logic Documentation (SC)
- Flow Diagrams (SC)

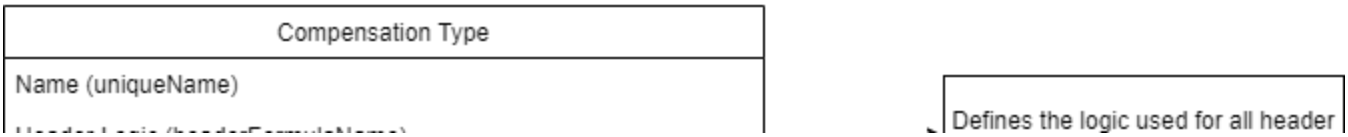
System Objects

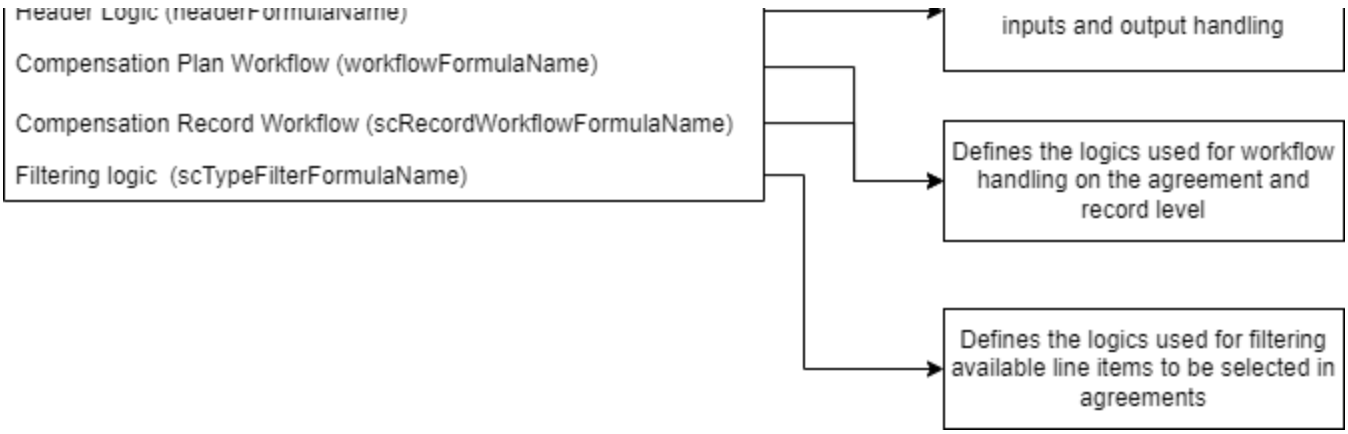

Seller Master Data
Seller Id (sellerId)
Seller Name (name)
Reports To (reportsTo)
First Name (attribute1)
Surname (attribute2)
Pricefx User Account Id (attribute3)
Active (attribute4)
Termination Date (attribute5)
Hire Date (attribute6)
Region (attribute7)
Market (attribute8)
Status (attribute9)
Used to identify the sellers. These fields are most often used in the code.
Used in dashboards to identify who is the manager of a given seller. If empty the person is the manager
Used in dashboards and selection options for more user friendly identification of sellers.
Used in dashboards to map a seller to a currently logged in user to restrict visible content
States whether the seller is active in the system.
Defines the termination date of the Seller, this affects generations and regenerations of records related to the seller
Currently not used


Compensation Record
ID (uniqueName)
Set (compensationRecordSetLabel)
Source ID (sourceId)
Source ID w/o revision (sourceIdNoRev)
Rank (rank)
Status (status)
Compensation Plan Status (compensationStatus)
Name (name)
Label (label)
Condition type (compensationConditionType)
Customer/Customer Grouping
Product/Product Grouping
Compensation Type (compensationHeaderType)
Payee Name (sellerName)
Defines what type of compensation has been selected
Taken from Plan (or Line Item) inputs or properties


Current Compensation (attribute2)
Pay To ID (attribute3)
Current Baseline Quantity (attribute4)
Forecasted Baseline Value (attribute8)
Compensation Forecast (attribute9)
True-up (attribute10)
Forecast Method (attribute11)
Sales Goal Increase % (attribute12)
Forecast Quantity (attribute14)
General Filter (attribute15)
Currency (attribute16)
Period Name (attribute17)
Previous Baseline Value (attribute18)
Previous Compensation (attribute19)
Current Period Adjustment Ids (attribute21)
Previous Period Adjustment Ids (attribute22)
Calculation results

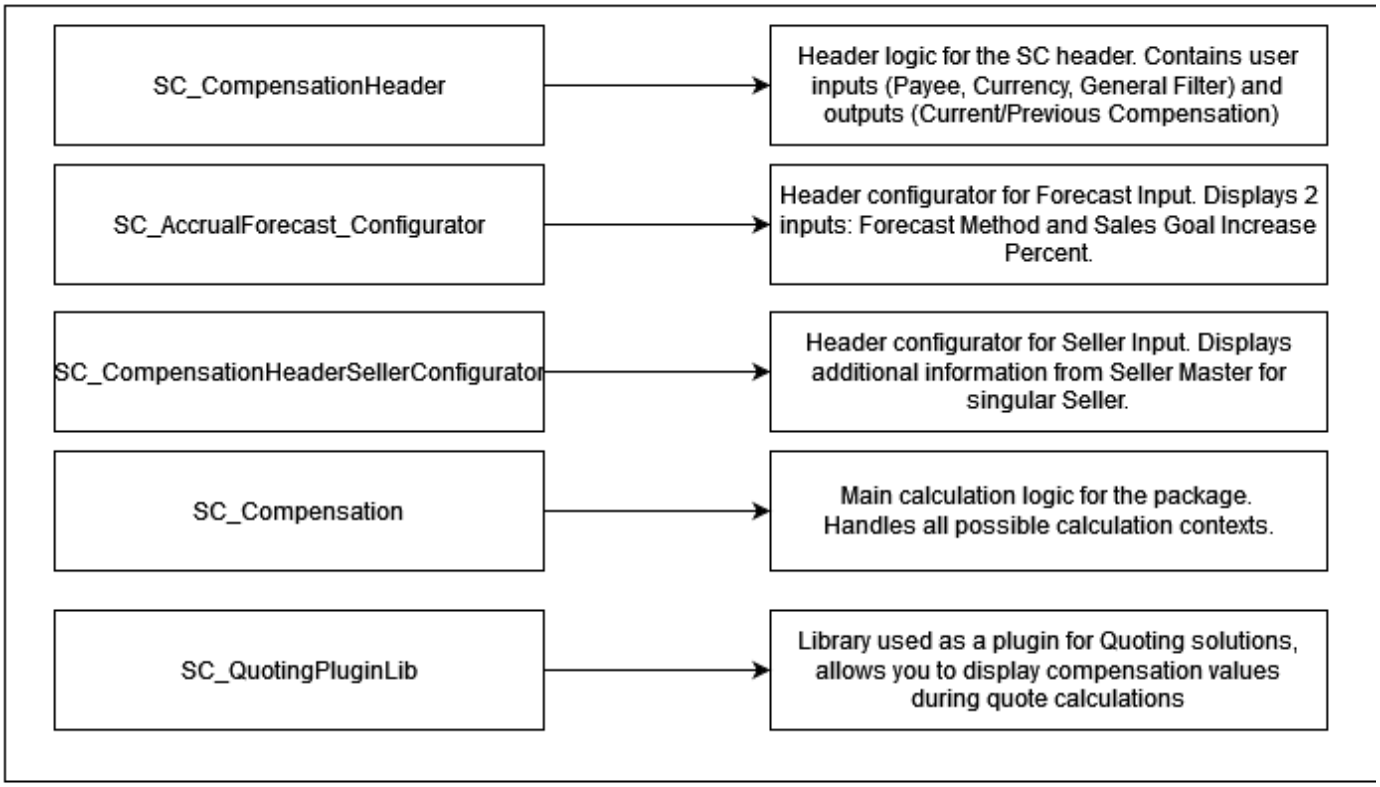

Compensation Plan
Name (uniqueName)
Name w/o revision (rootUniqueName)
Label (label)
Workflow Status (workflowStatus)
Workflow Submitter (submittedByName)
Customer(s) (customerGroup)
Payee Name (sellerName)
Status (compensationPlanStatus)
Compensation Type (compensationHeaderType)
Propagated from Header Input, defines the seller assigned to the agreement
Defines the type of the agreement. Used for filtering out non-SC related agreements, for example in dashboards


Compensation Type
Name (uniqueName)
Defines the logic used for all header

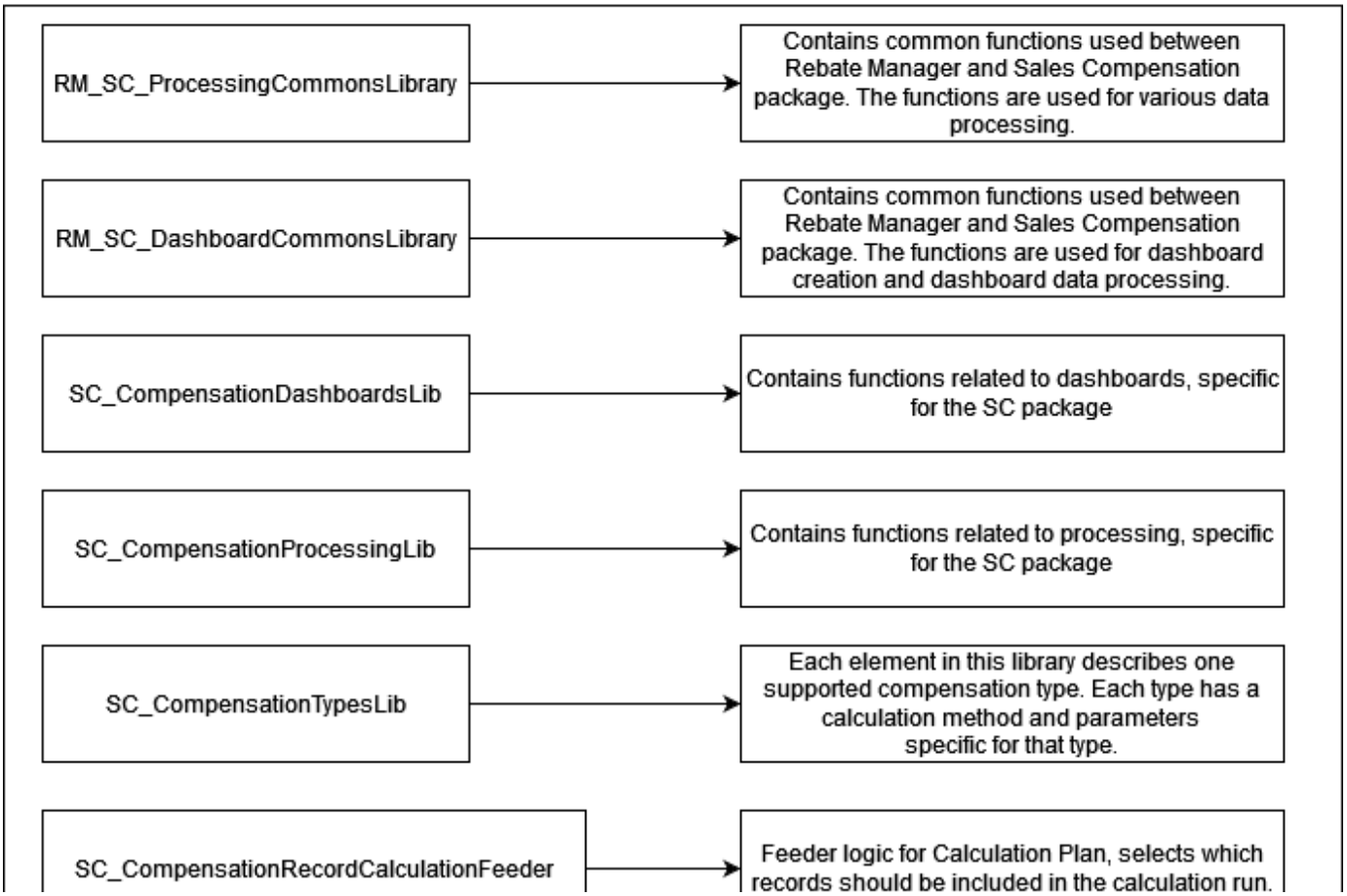

Header Logic (headerFormulaName)
Compensation Plan Workflow (workflowFormulaName)
Compensation Record Workflow (scRecordWorkflowFormulaName)
Filtering logic (scTypeFilterFormulaName)
inputs and output handling
Defines the logics used for workflow handling on the agreement and record level
Defines the logics used for filtering available line items to be selected in agreements

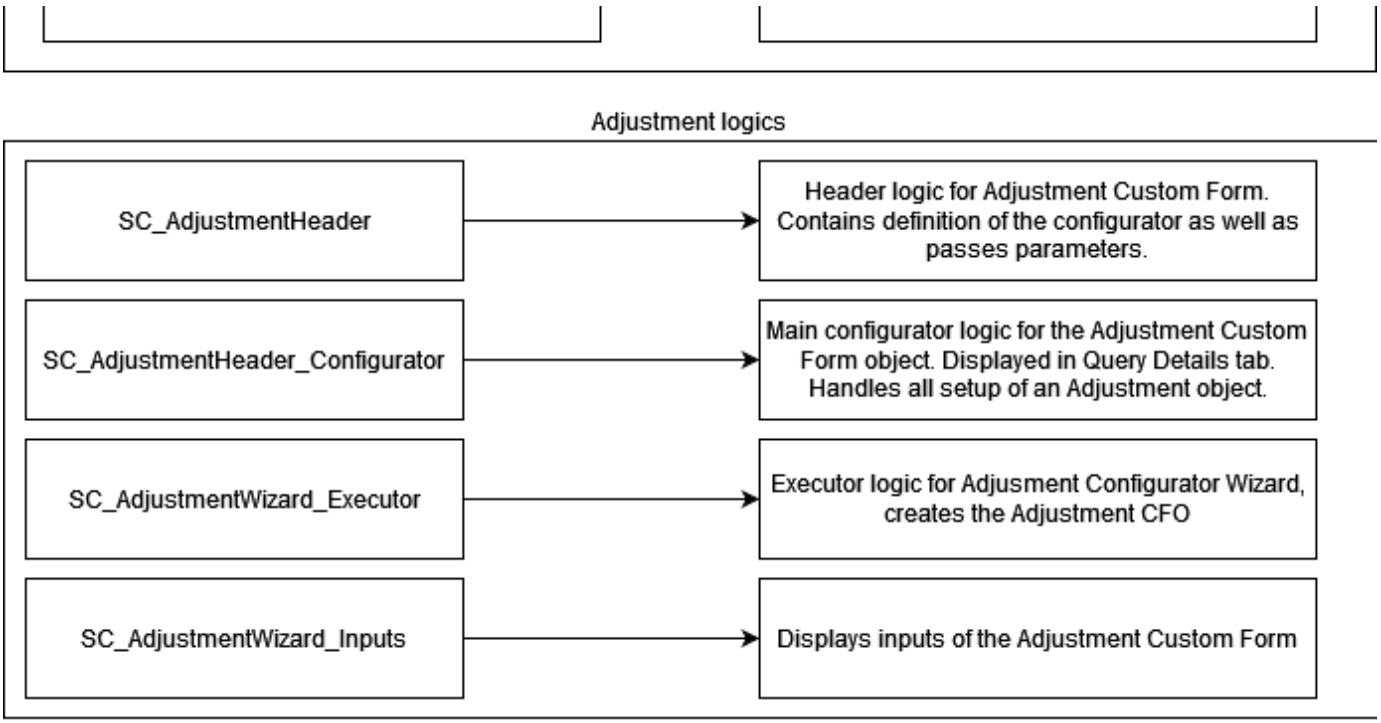
Logics

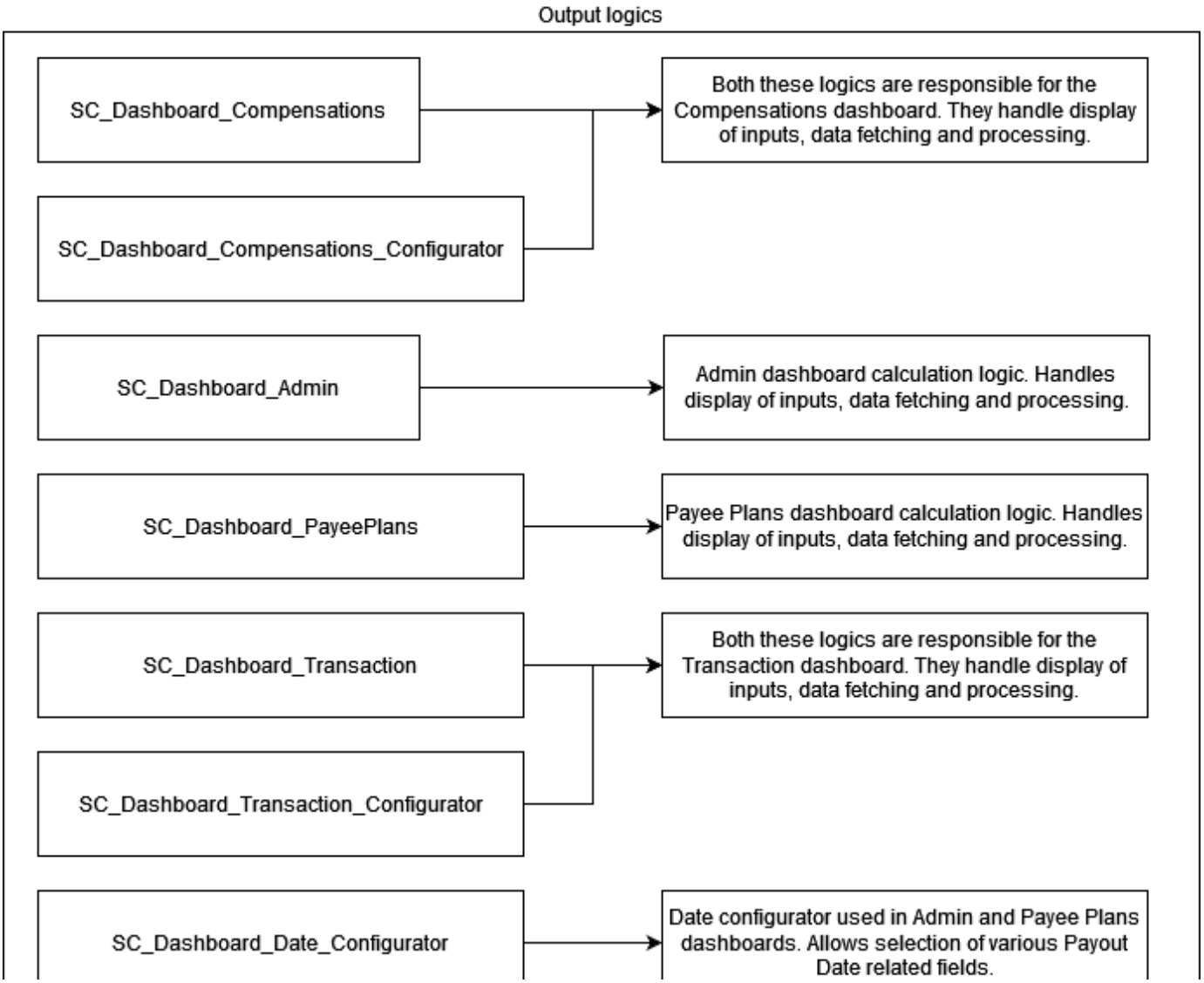
Core logics

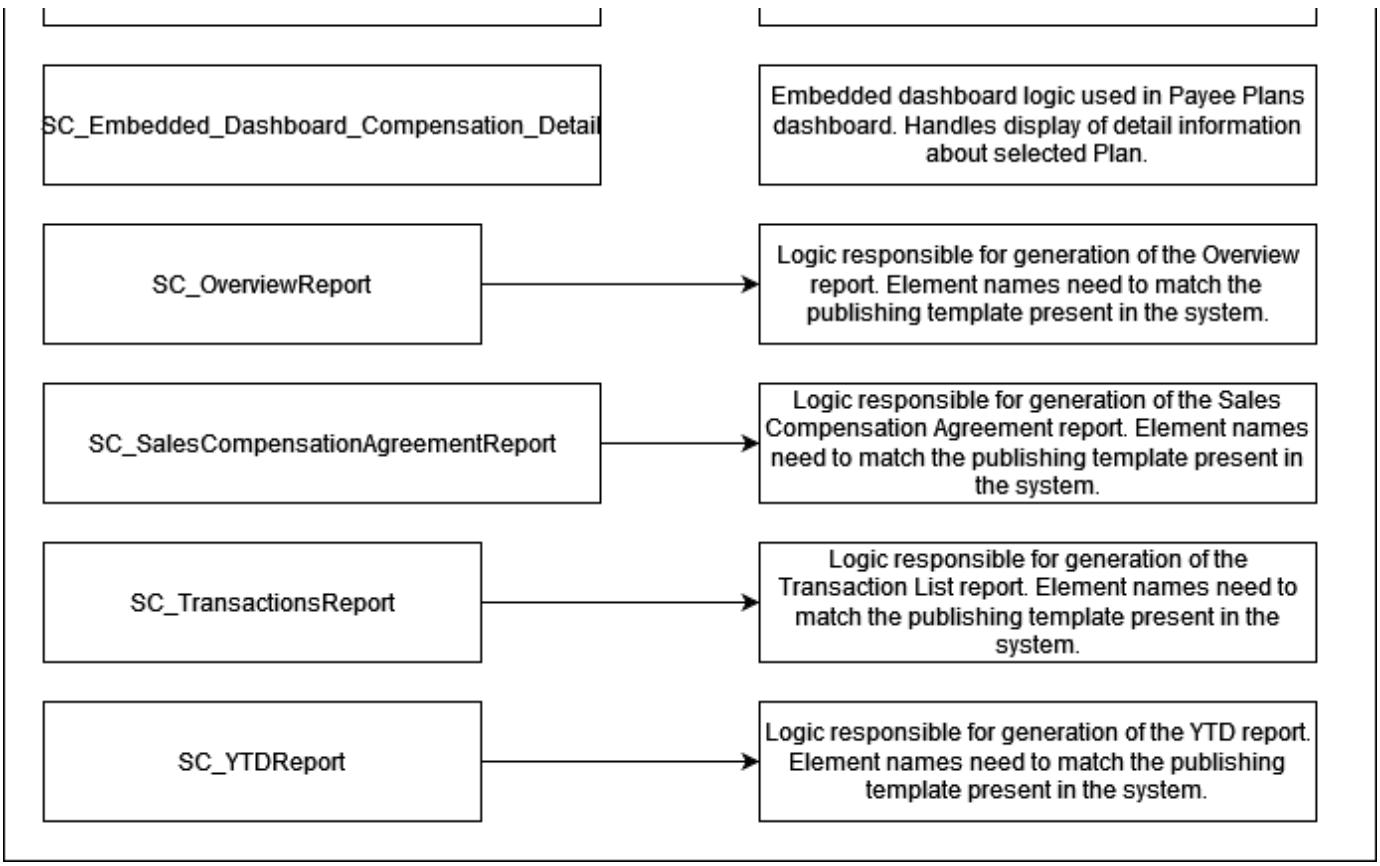


Backend logics

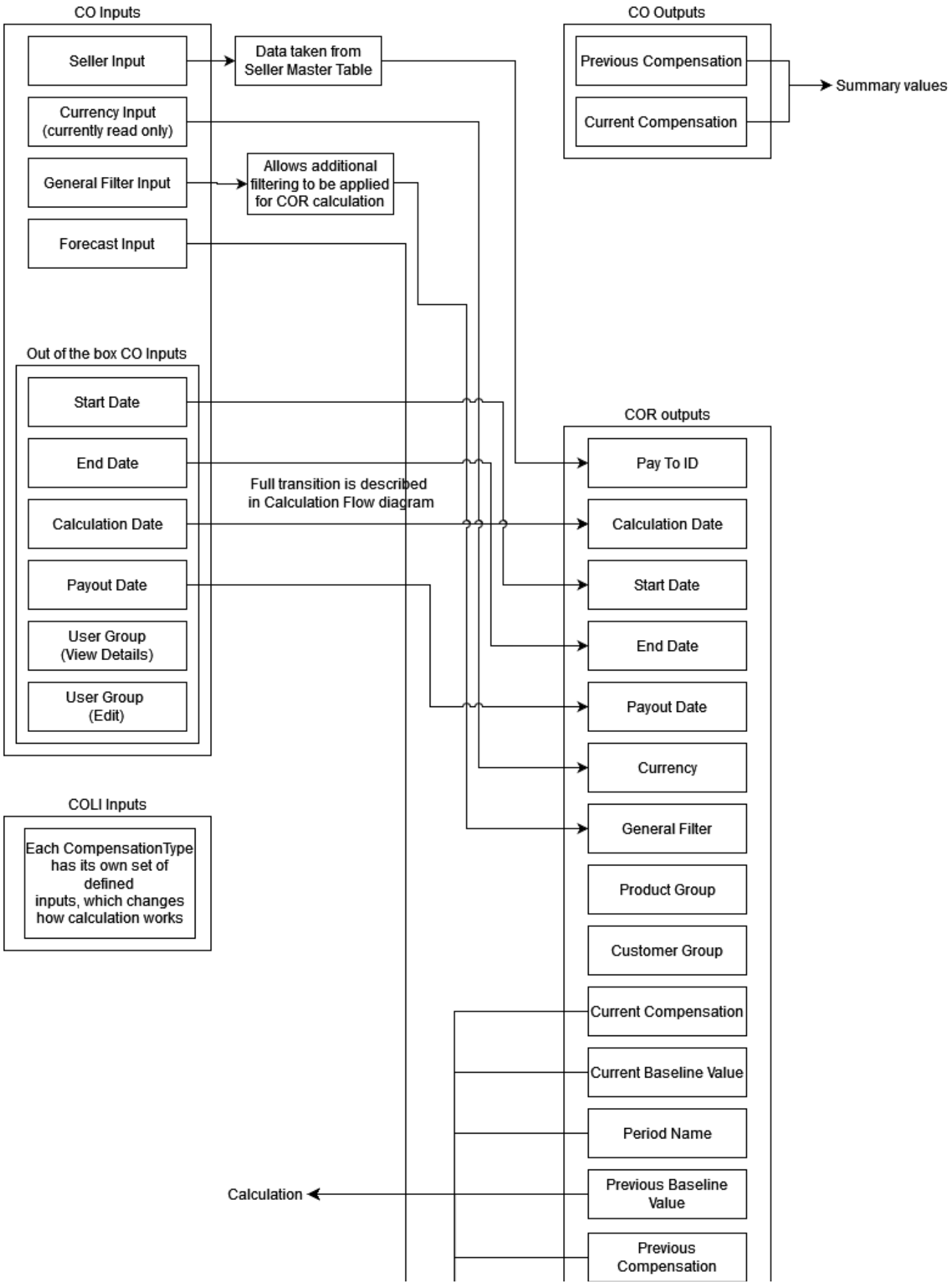


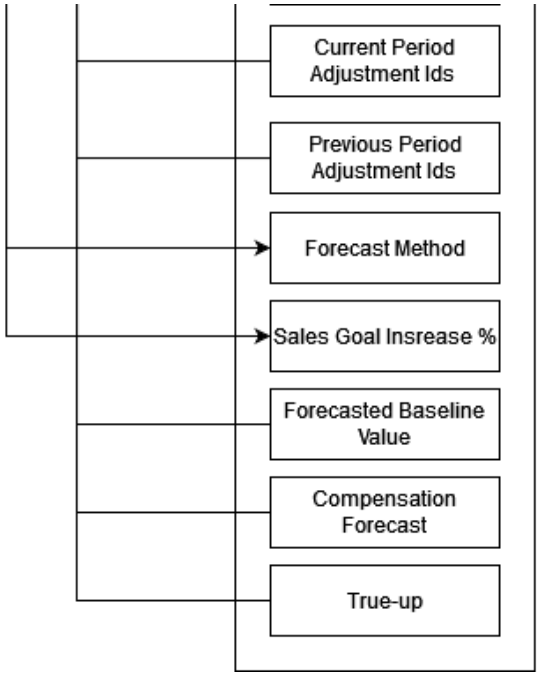

Adjustment logics
SC_AdjustmentHeader
Header logic for Adjustment Custom Form. Contains definition of the configurator as well as passes parameters.
SC_AdjustmentHeader_Configurator
Main configurator logic for the Adjustment Custom Form object. Displayed in Query Details tab. Handles all setup of an Adjustment object.
SC_AdjustmentWizard_Executor
Executor logic for Adjusment Configurator Wizard, creates the Adjustment CFO
SC_AdjustmentWizard_Inputs
Displays inputs of the Adjustment Custom Form


Output logics
SC_Dashboard_Compensations
Both these logics are responsible for the Compensations dashboard. They handle display of inputs, data fetching and processing.
SC_Dashboard_Compensations_Configurator
SC_Dashboard_Admin
Admin dashboard calculation logic. Handles display of inputs, data fetching and processing.
SC_Dashboard_PayeePlans
Payee Plans dashboard calculation logic. Handles display of inputs, data fetching and processing.
SC_Dashboard_Transaction
Both these logics are responsible for the Transaction dashboard. They handle display of inputs, data fetching and processing.
SC_Dashboard_Transaction_Configurator
SC_Dashboard_Date_Configurator
Date configurator used in Admin and Payee Plans dashboards. Allows selection of various Payout Date related fields.

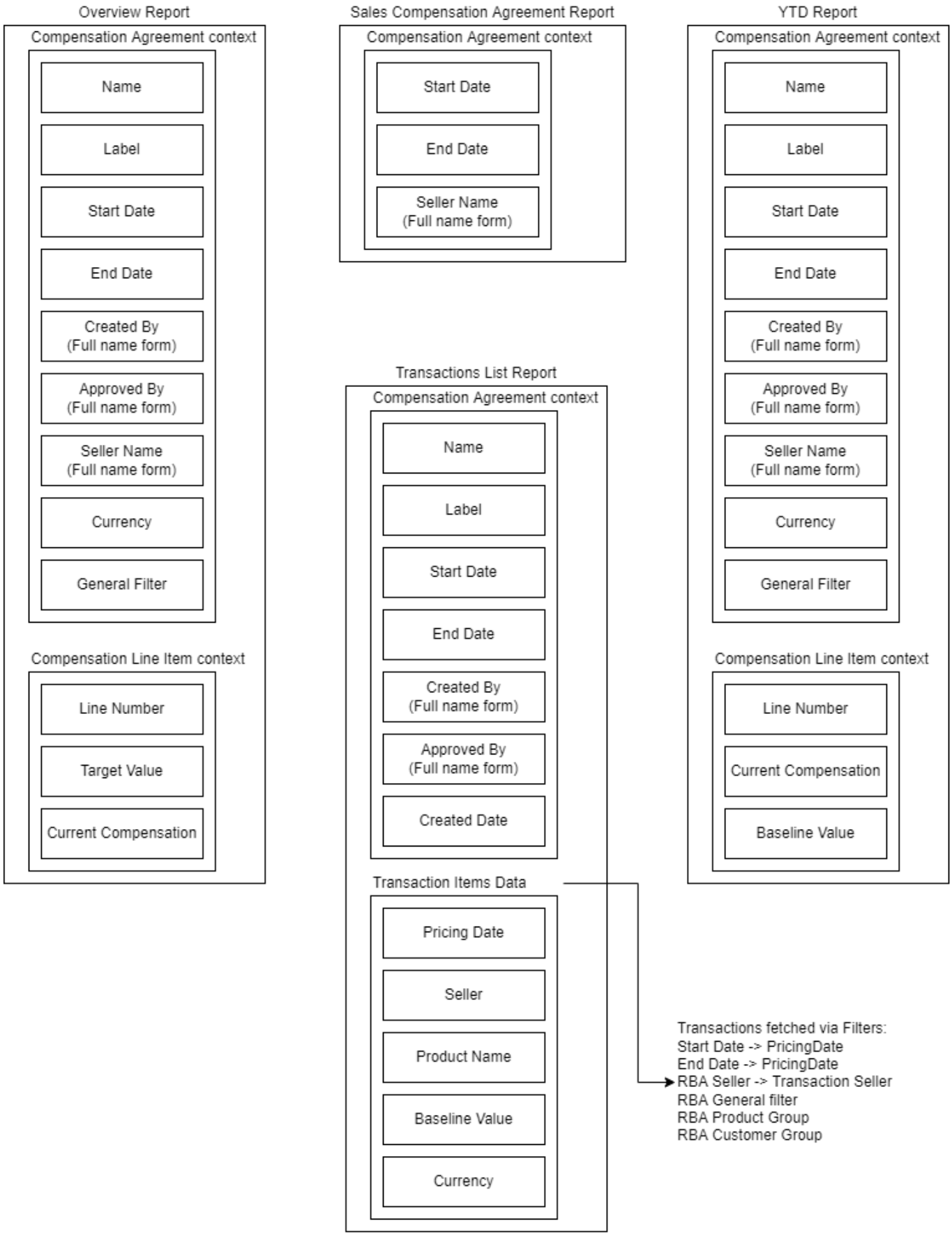

SC_Embedded_Dashboard_Compensation_Detail
Embedded dashboard logic used in Payee Plans dashboard. Handles display of detail information about selected Plan.
SC_OverviewReport
Logic responsible for generation of the Overview report. Element names need to match the publishing template present in the system.
SC_SalesCompensationAgreementReport
Logic responsible for generation of the Sales Compensation Agreement report. Element names need to match the publishing template present in the system.
SC_TransactionsReport
Logic responsible for generation of the Transaction List report. Element names need to match the publishing template present in the system.
SC_YTDReport
Logic responsible for generation of the YTD report. Element names need to match the publishing template present in the system.

Simplified Compensation Plan Compensation Record Data Flow


CO Inputs
Seller Input
Data taken from Seller Master Table
Currency Input (currently read only)
General Filter Input
Allows additional filtering to be applied for COR calculation
Forecast Input
CO Outputs
Previous Compensation
Current Compensation
Summary values
Out of the box CO Inputs
Start Date
End Date
Calculation Date
Payout Date
User Group (View Details)
User Group (Edit)
Full transition is described in Calculation Flow diagram
COR outputs
Pay To ID
Calculation Date
Start Date
End Date
Payout Date
Currency
General Filter
Product Group
Customer Group
Current Compensation
Current Baseline Value
Period Name
Previous Baseline Value
Previous Compensation
COLI Inputs
Each CompensationType has its own set of defined inputs, which changes how calculation works
Calculation


Current Period Adjustment Ids
Previous Period Adjustment Ids
Forecast Method
Sales Goal Insrease %
Forecasted Baseline Value
Compensation Forecast
True-up

Reports Data Flow



Logic Documentation (SC)

- RM_SC_ProcessingCommonsLibrary
- RM_SC_DashboardCommonsLibrary
- SC_CompensationDashboardsLib
- SC_CompensationProcessingLib
- SC_CompensationTypesLib
- SC_CompensationHeader
- SC_CompensationHeaderSellerConfigurator
- SC_AccrualForecast_Configurator
- SC_Compensation
- SC_Dashboard_Compensations_Configurator
- SC_Dashboard_Compensations
- SC_OverviewReport
- SC_SalesCompensationPlanReport
- SC_YTDReport
- SC_TransactionsReport
- SC_CompensationRecordCalculationFeeder
- SC_AdjustmentHeader
- SC_AdjustmentHeaderConfigurator
- SC_AdjustmentWizardExecutor
- SC_AdjustmentWizardInputs
- SC_QuotingPluginLib

RM_SC_ProcessingCommonsLibrary

Architecture Documentation

https://pricefx.atlassian.net/wiki/display/ACC/Architecture+Documentation+(SC)

Logic Description

This logic contains common utility functions that are used by both Rebate Manager and Sales Compensation packages. The library focuses on data processing related to the core of both those packages – that means calculation of plans and records mostly. The logic should be present if any of the two packages are in use.

Elements Description

- **ConstConfig** – Contains definitions of global constant variables that are used throughout the logic. Any configuration-like constants should be stored here to avoid hardcoding and to allow for better maintainability of the solution.
- **InputUtils** – Contains various methods that are used commonly when generating inputs. These inputs are mostly used for the line item level and are valid for various compensation and rebate types.
- **InputValidationUtils** – Defines methods used for input validation. This means common functions for error handling in case of invalid inputs as well as common constants (such as technical message separators).
- **DateUtils** – Contains methods related mostly to date and period definition. Reuses and extends a lot of methods from SharedLib.
- **FormatUtils** – Contains methods used for formatting output values to appropriate money, percentage etc. formats.
- **FilterUtils** – Handles the Product and Customer evaluated filters methods.

- **PeriodDataUtils** – Related to DateUtils, this util handles the Period structure creation and data access to it.
- **MathUtils** – Contains math related methods, such as percentage calculations.

RM_SC_DashboardCommonsLibrary

Architecture Documentation

https://pricefx.atlassian.net/wiki/display/ACC/Architecture+Documentation+(SC)

Logic Description

This logic contains common utility functions that are used by both Rebate Manager and Sales Compensation packages. The library focuses on dashboards and operations related to them – that is mostly data fetching and processing. The logic should be present if any of the two packages are in use.

Elements Description

- **ConstConfig** – Contains definitions of global constant variables that are used throughout the logic. Any configuration-like constants should be stored here to avoid hardcoding and to allow for better maintainability of the solution.
- **ConfigurationUtils** – Contains common methods related to Advanced Configuration handling, mostly fetching and creating appropriate structures. The way the Advanced Configuration is loaded is defined by each package separately.
- **InputUtils** – Contains methods related to common user dashboard inputs.
- **FormatUtils** – Contains methods used for formatting output values to appropriate money, percentage etc. formats. Currency symbol generation methods are also stored here.
- **PaybackTypeUtils** – Holds methods related to fetching and processing system data for various types of payback (rebate/compensation).
- **PaybackAgreementUtils** – Contains everything related to payback agreements (rebate/compensation), holds methods for generation of data structures as well as a way to fetch the data and display it.

SC_CompensationDashboardsLib

Architecture Documentation

https://pricefx.atlassian.net/wiki/display/ACC/Architecture+Documentation+(SC)

Logic Description

This library extends the RM_SC_DashboardCommonsLibrary with Sales Compensation specific elements and methods.

Elements Description

- **ConfigurationUtils** – Extension of the common element from RM_SC_DashboardCommonsLibrary, contains SC package specific configuration structures and loading operations. This is the place where all Advanced Configuration configuration fetching and storage takes place.
- **ConstConfig** – Contains definitions of global constant variables that are used throughout the logic. Any configuration-like constants should be stored here to avoid hardcoding and to allow for better maintainability of the solution.
- **CompensationTypeUtils** – Contains methods related to fetching and accessing data from Compensation Type object (COCT).

- **CompensationRecordUtils** - Contains methods related to fetching and accessing data from Compensation Record object (COR). This util also defines how the Compensation Record data are structured.
- **CompensationAgreementUtils** - Contains methods related to fetching data from Compensation Plan object (CO).
- **CompensationAgreementLineItemUtils** - Defines methods for accessing and building the structures for Compensation Plan Line Items ("COLI" currently). The data is extracted from the already fetched and processed Compensation Plan objects.
- **ConfiguratorUtils** - Contains methods used in various configurators for dashboards. These are mostly input generation methods.
- **DateUtils** - Contains methods that relate to date in any way, such as filter generation for years or start /end dates.
- **InputValidationUtils** - Contains methods used for input validation in dashboards - such as checking if input is not empty or has valid values.
- **PortletUtils** - Contains methods that generate ready to use portlets, such as summary portlets.
- **QueryUtils** - Contains methods related to any type of Datamart / Data Source lookups and queries.
- **CurrencyUtils** - Contains methods related to currency (formats, fetching or label generation).

SC_CompensationProcessingLib

Architecture Documentation

https://pricefx.atlassian.net/wiki/display/ACC/Architecture+Documentation+(SC).

Logic Description

This library extends the RM_SC_ProcessingCommonsLibrary with Sales Compensation specific elements and methods. The most important elements focus on calculation of specific SC package Compensation Types, as well as handling the SC specific data objects (fetching, processing).

Elements Description

- **ConstConfig** - Contains definitions of global constant variables that are used throughout the logic. Any configuration-like constants should be stored here to avoid hardcoding and to allow for better maintainability of the solution.
- **ConfigManager** - Strictly defined structure that allows management of solution configuration. The benefit is that the whole configuration is contained within one "class", it cannot be modified and is cached.
- **CompensationInputParameter** - "Class" like element that provides a creation method for a structure called InputParameter which stores all user inputs for further processing or access.
- **InputUtils** - Extension of RM_SC_ProcessingCommonsLibrary InputUtils element with Sales Compensation related methods. Contains mostly methods for creating inputs on line item level.
- **InputValidationUtils** - Extension of RM_SC_ProcessingCommonsLibrary InputValidationUtils, contains checks for Sales Compensation specific input types and their validation.
- **QueryUtils** - Contains methods related to Datamart queries, mostly for transaction type data.
- **FormatUtils** - Extension of RM_SC_ProcessingCommonsLibrary FormatUtils, extends the formatting by coloring options.
- **CalculationUtils** - Contains methods for final steps of the core logic, creation of Compensation Records and enrichment of Compensation Record data based on the calculation base.
- **CacheDataManagementUtils** - Contains methods for various caching purposes.
- **CompensationDataCalculator** - Contains methods related to main calculations, that is for example calculation of current and previous compensation in various contexts. Calculations related to transaction data can also be found here.

- **CompensationAgreementUtils** - Extension of RM_SC_ProcessingCommonsLibrary PaybackAgreementUtils with Sales Compensation related methods. The main addition is the focus on Compensation Agreement objects as well as fetching and processing data coming from those objects.
- **BasicCompensationValuesDataUtils** - Helper util element that supports compensation calculations, summaries etc.
- **CompensationCalculationParameter** - "Class" like element that provides a creation method for a structure called CalculationParameter which stores all calculation related closures for further processing or access.
- **CompensationValueCalculator** - Contains methods that describe the calculation methods for various types of compensations: target, stepped, growth or multi.
- **FilterUtils** - Extension of RM_SC_ProcessingCommonsLibrary FilterUtils with Sales Compensation related methods. Mostly related to building a product/customer filter structure.
- **CompensationRecordUtils** - Extension of RM_SC_ProcessingCommonsLibrary PaybackRecordUtils with Sales Compensation related methods. Defines a way of fetching the CompensationRecords as well as provides a way of building filters for the fetch based on the provided data.
- **ReportUtils** - Contains utils that are used in all reports in the package. This util includes data fetches for reports as well as generation of report specific labels.
- **SellerUtils** - Contains methods that are used for all Seller related data manipulation - that is fetching, retrieval based on some specific fields etc.
- **AdjustmentUtils** - Contains methods that are related to Adjustment Custom From, their creation, processing and access of data.
- **AdjustmentConfiguratorUtils** - Contains methods used in Adjustment related configurators (Adjustment Form and Adjustment Creation Wizard) and mainly consists of input generation methods.
- **AdjustmentValidationUtils** - Contains methods related to validation of Adjustment inputs.
- **CompensationTypeUtils** - Contains methods related to CompensationTypes. Provides methods for easy access to all necessary information of a given CompensationType, as well as processing methods for its data such as reading Target or Compensation values from line items.
- **AccrualForecastUtils** - Contains calculation methods related to Forecasts.
- **ChartUtils** - Contains methods that generate charts. These charts are displayed in the outputs of the Compensation Plan, hence they have been separated from the SC_CompensationDashboardsLib.

SC_CompensationTypesLib

Architecture Documentation

https://pricefx.atlassian.net/wiki/display/ACC/Architecture+Documentation+(SC)

Logic Description

The library stores all types of currently supported Compensation Types that can be added as line items in Compensation Agreements. Each type has a strictly defined structure: Calculation Type, Target Type, Target Value Type, Compensation Value Type as well as a way to calculate the compensation. The element names match 1-1 the Condition Types defined in the system and should be carefully maintained if any changes arise.

Elements Description

- SingleTargetAmount
- SingleTargetPercent
- MultiTargetAmount
- MultiTargetPercent
- SteppedAmount
- SteppedPercent
- GrowthAbsoluteAmount

- GrowthAbsolutePercent
- GrowthPercentAmount
- GrowthPercentPercent

SC_CompensationHeader

Architecture Documentation

https://pricefx.atlassian.net/wiki/display/ACC/Architecture+Documentation+(SC)

Logic Description

This is currently the only header logic that defines the header behavior of Compensation Plan. The logic defines the inputs and outputs from the header to further operations that happen in the main core logic.

Elements Description

- **ConfigManager** - Initialization of a ConfigManager structure defined in SC_CompensationProcessingLib.
- **HeaderUtils** - Contains methods for creating header inputs and outputs, main codebase of the logic. Calculation methods for Previous and Current compensation can also be found here.
- **SyntaxCheckAbort**
- **HeaderInputs** - Contains initialization of all header inputs using HeaderUtils (even the hidden ones, such as the type of the header).
- **HeaderOutputs** - Adds outputs to the header using HeaderUtils.

SC_CompensationHeaderSellerConfigurator

Architecture Documentation

https://pricefx.atlassian.net/wiki/display/ACC/Architecture+Documentation+(SC)

Logic Description

This is the configurator for Seller selection on the header level. It is used only in header types that contain a singular Seller. It displays additional information that is configured to be displayed in SC_HeaderSellerDetailsMapping PP table.

Elements Description

- **SellerInput** - Returns a user input that allows a Seller selection.
- **SellerDetailsInput** - Returns a read only user input that, based on selection in SellerInput, displays additional Seller information provided in the SC_HeaderSellerDetailsMapping PP table.

SC_AccrualForecast_Configurator

Architecture Documentation

https://pricefx.atlassian.net/wiki/display/ACC/Architecture+Documentation+(SC)

Logic Description

This is the configurator for the Forecast input on the header level. It is used to display forecast inputs for the user.

Elements Description

- **AccrualForecastMethod** – Generates the Forecast Method options selection input that allows the user to provide forecast method.
- **SalesGoalIncreasePercent** – Presents sales goal increase % value user input.
- **PreviousAccrualForecastMethod** – Stores the old value of AccrualForecastMethod. It is used to check the current value of AccrualForecastMethod to reset value for SalesGoalIncreasePercent.

SC_Compensation

Architecture Documentation

https://pricefx.atlassian.net/wiki/display/ACC/Architecture+Documentation+(SC)

Logic Description

This is the core logic for the whole package. The logic works in four contexts:

1. *Agreement* context (line items) which as a result outputs a set of Compensation Records with some data prefilled.
2. *Generate Compensation Record* context which regenerates records if any of the main source data changed.
3. *Generate Accrual Records* context which constructs the snapshot of the current transaction data and stores them in Accrual Records to be used in calculations later on.
4. *Compensation Record* which does the actual calculation that is run via the Sales Compensation Calculation object on a set schedule.

This logic – due to its size – has "context separators" which make it easier to see which elements run in which context. The separators are empty elements with the following name pattern: `____SOME_NAME_CONTEXT1_CONTEXT2____`

- `SOME_NAME` describes what is in the section.
- `CONTEXT1`, `CONTEXT2` define the contexts this section runs in.

Elements Description

- **CompensationConfig** – Initialization of a ConfigManager structure defined in SC_CompensationProcessingLib.
- **CompensationTypeCode** – Retrieves the type name of the currently processed Compensation Line Item. This is the Name column value from ConditionTypes. This code is used to access the appropriate element from the CompensationTypesLib later on.
- **CalculationType** – Defines the calculation type of a currently processed Compensation Type. Can be one of these values: Conditional, Growth, Stepped.
- **TargetType** – Defines the target type of a currently processed Compensation Type. Can be one of these values: None, Single, Multi.
- **TargetValueType** – Defines the target value type of a currently processed Compensation Type. Can be one of these values: Amount, Percent.
- **CompensationValueType** – Defines the compensation value type of a currently processed Compensation Type. Can be one of these values: Amount, Amount Per Unit, Percent.
- **DatamartCurrency** – Defines the currency used by the main Datamart. The Datamart configuration is stored in SC_AdvancedConfiguration.
- **CustomerGroupInput** – User input for Customer Group, used for filtering later on.
- **ProductGroupInput** – User input for Product Group, used for filtering later on.

- **SingleTargetInput** – Returns a user input for a particular Compensation Type that has the target defined as Single. Allows input of a target factor.
- **SingleCompensationInput** – Returns a user input for a particular Compensation Type that has the target defined as Single. Allows input of a compensation value.
- **MultiTargetInput** – Returns a user input for a particular Compensation Type that has the target defined as Multi and Calculation Type as one of Conditional, Growth or Stepped. Allows input of a combination of both target factor and compensation value in a multitiered fashion.
- **PaymentPeriodInput** – Displays a used input for a Payment Period selection as an option. Available values are Monthly, Quarterly, Semi-Annually, Annually.
- **QuotaForInput** – Displays a read only user input containing the QuotaFor value stored in Condition Types table. If the value is not present, the value from Advanced Configuration SC_AdvancedConfiguration is taken instead.
- **DepositSchemeInput** – Displays a read only user input containing the DepositScheme value stored in Condition Types table. If the value is not present, the value from Advanced Configuration SC_AdvancedConfiguration is taken instead.
- **BaselineFieldNameInput** – Displays a read only user input containing the BaselineFieldName value stored in Condition Types table. If the value is not present, the value from Advanced Configuration SC_AdvancedConfiguration is taken instead.
- **SyntaxCheckAbort**
- **HeaderSellerGroupInput** – Retrieves the Seller group input value from the header level.
- **HeaderCurrencyInput** – Retrieves the currency value from the header level.
- **HeaderTypeInput** – Retrieves the type of the header.
- **ObjectValidation** – Performs validation of the state of a currently processed item in order to assess whether the logic execution should continue.
- **SellerIdsInput** – Returns the Seller group converted into a list of individual Seller IDs.
- **CompensationInputDefinition** – Returns CompensationInputParameter that stores all the inputs taken from user/header etc. This parameter is used later on for other processing/access operations.
- **InputValidation** – Performs validation on all inputs that were provided. If any input is marked as invalid, an exception is thrown with a proper message to the user.
- **PreviousPeriodAdjustmentData** – Returns the previous period adjustment data used later on in calculations.

  CurrentPeriodAdjustmentData – Returns the current period adjustment data used later on in calculations.
- **PreviousPeriodData** – Returns a value of the previous period compensation based on the current context.
- **CurrentPeriodData** – Returns a value of the current period compensation based on the current context.
- **BaselineValueFieldLabel** – Returns a field in a Datamart that will be used for baseline calculations. The field is defined in the Datamart Advanced Options configuration.
- **PreviousBaselineValue** – Extracts the baseline value from the previous period data fetched in one of the previous elements.
- **CurrentBaselineValue** – Extracts the baseline value from the current period data fetched in one of the previous elements.
- **PreviousCompensation** – Extracts the compensation value from the previous period data fetched in one of the previous elements. Colors the result according to the value.
- **CurrentCompensation** – Extracts the baseline value from the current period data fetched in one of the previous elements. Colors the result according to the value.
- **ForecastMethod** – Passes the value of Forecast Method input.
- **SalesGoalIncreasePct** – Passes the value of Sales Goal Increase % input.
- **AccrualForecastBaselineValue** – Extracts the forecasted baseline value from the current period data fetched in one of the previous elements.
- **AccrualForecastCompensation** – Extracts the forecasted compensation value from the current period data fetched in one of the previous elements.

- **TrueUp** – Accrual Forecast Compensation minus Current Compensation.
- **TargetFor** – Retrieves the target for a value stored in the configuration of a currently processed Compensation Type of Advanced Configuration of the package (default). Can be either Payment Period or Annual.
- **DepositScheme** – Retrieves the deposit scheme value stored in the configuration of a currently processed Compensation Type of Advanced Configuration of the package (default). Can be either Cumulative or Non-Cumulative.
- **PaymentPeriod** – Passes the value of Payment Period input.
- **SellerNamesTable** – Passes the value of Seller input, stores the name of the Sellers.
- **CustomerGroup** – Passes the value of the CustomerGroup input.
- **ProductGroup** – Passes the value of the ProductGroup input.
- **GeneralFilter** – Passes the value of the GeneralFilter input.
- **Currency** – Passes the value of the Currency input.
- **CurrentPeriods** – Returns a list of periods based on the line item payment period input and CO plan startDate/endDate.
- **PreviousPeriods** – Returns a list of previous periods based on the line item payment period input and CO plan startDate/endDate.
- **PreviousCompensationDataOnPeriods** – Returns data (adjustment data & Datamart data) of the previous periods. The period is based on the payment period input, used later on in calculations.
- **CurrentCompensationDataOnPeriods** – Returns data (adjustment data & Datamart data) of the current periods. The period is based on the payment period input, used later on in calculations.
- **PreviousMonthlyPeriodsData** – Returns data (adjustment data & Datamart data) of the previous MONTHLY periods, used to aggregate data in the header.
- **CurrentMonthlyPeriodsData** – Returns data (adjustment data & Datamart data) of the current MONTHLY periods, used to aggregate data in the header.
- **AddingMonthlyPeriodsDataToCache** – Adds previous monthly periods data and current monthly periods data to the cache, used to pass data to the header (post phase) for aggregation header data.
- **AccrualCompensationComparisonChart** – Bar chart displaying Previous Compensation vs. Current Compensation vs. Forecasted Compensation.
- **AccrualForecastBaselineValueTable** – Table displaying this year's sales forecast by Payment Period of the particular line item.
- **AccrualForecastBaselineValueChart** – Bar chart displaying this year's sales forecast by Payment Period of the particular line item.
- **PreviousPeriodAdjustmentIds** – Returns the list of IDs of Adjustments that were taken into account during calculations. These Adjustments are taken from the previous period.
- **CurrentPeriodAdjustmentIds** – Returns the list of IDs of Adjustments that were taken into account during calculations. These Adjustments are taken from the current period.
- **CompensationRecords** – Generates the Compensation Record shells according to the processed information. This element is only run in the Plan context.
- **RegenerateCompensationRecords** – Re-generates the Compensation Records if any of the re-generation conditions were met. This element is run only in the Generate Compensation Record context.
- **GenerateAccrualRecords** – Generates the main calculation data snapshot in the Accrual Records table. This element is run only in the Generate Accrual Record context.
- **CalculationBase** – Returns the calculationBase for the Compensation Record.

SC_Dashboard_Compensations_Configurator

Architecture Documentation

https://pricefx.atlassian.net/wiki/display/ACC/Architecture+Documentation+(SC)

Logic Description

This is the configurator logic for the Compensations Dashboard. Its main purpose is to allow selection of different Compensation Agreements based on the selected Seller.

Elements Description

- **SellerInput** - Presents the user with a Seller input if the user is a Sales Manager (allows to select one of the Sellers out of all who report to that person) or if the user is not a Sales Manager (a text with what Seller is being used for further processing).
- **CompensationInput** - Provides a Compensation Agreement selection option based on the Seller input defined above. The agreements are fetched in YTD period and only approved ones are considered.

SC_Dashboard_Compensations

Architecture Documentation

https://pricefx.atlassian.net/wiki/display/ACC/Architecture+Documentation+(SC)

Logic Description

This is a logic for the Compensation Dashboard. It handles display of all inputs, validation of the inputs, data fetching as well as processing of the obtained data. The dashboard displays three portlets: one data table and two charts.

Elements Description

- **Configuration** - Fetches the Advanced Configuration options for the package. The most important part for the dashboard is the Datamart configuration.
- **DashboardUtils** - Contains all methods related to creation of portlets and data processing related to them. The utils are for all three currently visible portlets.
- **StartDate** - Stores the start date of the current year for further processing.
- **EndDate** - Stores the end date of the current month for further processing.
- **Configurator** - Displays the inline configurator defined by SC_Dashboard_Compensations_Configurator logic.
- **TimePeriod** - Allows selection of time period that modifies the results returned by the data fetch.
- **AbortIfIsSyntaxCheck**
- **InputValidation** - Validates any user input. If it is incorrect, an exception is thrown. Currently only Seller input is validated.
- **DatamartCurrency** - Returns the currency in which the main transaction Datamart stores data. The Datamart is defined in the Configuration element (Advanced Configuration).
- **CompensationAgreementData** - Main data fetch element for the dashboard, it retrieves Datamart data based on the provided filters such as time period, Seller etc. The data is returned in structures defined by the PaybackAgreementUtils element in the RM_SC_DashboardCommonsLibrary library.
- **CompensationAgreementTable** - Performs processing operations on the data returned from the eCompensationAgreementData element in order to display the defined data table portlet. The data table rows/columns are created in this element.
- **CompensationAgreementByUniqueNameChart** - Gathers information for the display and displays the CompensationAgreementByUniqueNameChart portlet.
- **CompensationAgreementByTimePeriodChart** - Gathers information for the display and displays the CompensationAgreementByTimePeriodChart portlet.

SC_OverviewReport

Architecture Documentation

https://pricefx.atlassian.net/wiki/display/ACC/Architecture+Documentation+(SC)

Logic Description

This is a logic for the Overview Report. The logic is strictly tied to a publishing template present also in the repository. The element names that are used for data display need to match 1-1 in order to display properly.

Elements Description

- **CompensationAgreementData** - Returns the data of the currently open Compensation Agreement.
- **AgreementName** - Display element, returns the agreement name.
- **Label** - Display element, returns the agreement label.
- **StartDate** - Display element, returns the agreement start date.
- **EndDate** - Display element, returns the agreement end date.
- **CreatedByPrimary** - Display element, returns the Created By primary part of the Created By name. The primary part contains either name and surname of the Seller if defined or the system username if the Seller is not found.
- **CreatedBySecondary** - Display element, returns the secondary part of the Created By name. The secondary part is displayed only if the name and surname of the Seller are known and contain a system username in brackets.
- **CreatedDate** - Display element, returns the creation date of the agreement.
- **ApprovedByPrimary** - Display element, returns the Created By primary part of the Approved By name. The primary part contains either name and surname of the Seller if defined or the system username if the Seller is not found.
- **ApprovedBySecondary** - Display element, returns the secondary part of the Approved By name. The secondary part is displayed only if the name and surname of the Seller are known and contain a system username in brackets.
- **PrimarySellerName** - Display element, returns the Created By primary part of the Seller name. The primary part contains either name and surname of the Seller if defined or the system username if the Seller is not found.
- **SecondarySellerName** - Display element, returns the secondary part of the Seller name. The secondary part is displayed only if the name and surname of the Seller are known and contain a system username in brackets.
- **Currency** - Display element, returns the currency of the agreement.
- **GeneralFilter** - Display element, returns the value of the general filter input that is present on the agreement.
- **CompensationItems** - Returns a list of all line items present in the agreement. The line items have to be returned as a list of maps where each map reflects one line item value, the structure is clearly defined and has to map 1-1 with what is in the publishing template.
- **TotalCompensation** - Display element, returns the sum of total compensations of all agreement line items.

SC_SalesCompensationPlanReport

Architecture Documentation

https://pricefx.atlassian.net/wiki/display/ACC/Architecture+Documentation+(SC)

Logic Description

This is a logic for the Sales Compensation Agreement Report. The logic is strictly tied to a publishing template present also in the repository. The element names that are used for data display need to match 1-1 in order to display properly.

Elements Description

- **CompensationAgreementData** – Returns the data of the currently open Compensation Agreement.
- **StartDate** – Display element, returns the agreement start date.
- **EndDate** – Display element, returns the agreement end date.
- **PrimarySellerName** – Display element, returns the Created By primary part of the Seller name. The primary part contains either name and surname of the Seller if defined or the system username if the Seller is not found.
- **SecondarySellerName** – Display element, returns the secondary part of the Seller name. The secondary part is displayed only if the name and surname of the Seller are known and contain a system username in brackets.

SC_YTDReport

Architecture Documentation

https://pricefx.atlassian.net/wiki/display/ACC/Architecture+Documentation+(SC)

Logic Description

This is a logic for the YTD Report. The logic is strictly tied to a publishing template present also in the repository. The element names that are used for data display need to match 1-1 in order to display properly.

Elements Description

- **CompensationAgreementData** – Returns the data of the currently open Compensation Agreement.
- **AgreementName** – Display element, returns the agreement name.
- **Label** – Display element, returns the agreement label.
- **ReportStartDate** – Stores the start date of the current year for further processing.
- **ReportEndDate** – Stores the end date of the current month for further processing.
- **ReportCurrentDate** – Stores the current date for further processing.
- **CreatedByPrimary** – Display element, returns the Created By primary part of the Created By name. The primary part contains either name and surname of the Seller if defined or the system username if the Seller is not found.
- **CreatedBySecondary** – Display element, returns the secondary part of the Created By name. The secondary part is displayed only if the name and surname of the Seller are known and contain a system username in brackets.
- **ApprovedByPrimary** – Display element, returns the Created By primary part of the Approved By name. The primary part contains either name and surname of the Seller if defined or the system username if the Seller is not found.
- **ApprovedBySecondary** – Display element, returns the secondary part of the Approved By name. The secondary part is displayed only if the name and surname of the Seller are known and contain a system username in brackets.
- **PrimarySellerName** – Display element, returns the Created By primary part of the Seller name. The primary part contains either name and surname of the Seller if defined or the system username if the Seller is not found.

- **SecondarySellerName** – Display element, returns the secondary part of the Seller name. The secondary part is displayed only if the name and surname of the Seller are known and contain a system username in brackets.
- **Currency** – Display element, returns the currency of the agreement.
- **GeneralFilter** – Display element, returns the value of the general filter input that is present on the agreement.
- **CompensationItems** – Returns a list of all line items present in the agreement. The line items have to be returned as a list of maps where each map reflects one line item value, the structure is clearly defined and has to map 1-1 with what is in the publishing template.
- **TotalCompensation** – Display element, returns the sum of total compensations of all agreement line items.

SC_TransactionsReport

Architecture Documentation

https://pricefx.atlassian.net/wiki/display/ACC/Architecture+Documentation+(SC)

Logic Description

This is a logic for the YTD Report. The logic is strictly tied to a publishing template present also in the repository. The element names that are used for data display need to match 1-1 in order to display properly.

Elements Description

- **ConfigManager** – Provides an instance of ConfigManager of the package.
- **CompensationAgreementData** – Returns the data of the currently open Compensation Agreement.
- **AgreementName** – Display element, returns the agreement name.
- **Label** – Display element, returns the agreement label.
- **CreatedByPrimary** – Display element, returns the Created By primary part of the Created By name. The primary part contains either name and surname of the Seller if defined or the system username if the Seller is not found.
- **CreatedBySecondary** – Display element, returns the secondary part of the Created By name. The secondary part is displayed only if the name and surname of the Seller are known and contain a system username in brackets.
- **CreatedDate** – Display element, returns the creation date of the agreement.
- **ApprovedByPrimary** – Display element, returns the Created By primary part of the Approved By name. The primary part contains either name and surname of the Seller if defined or the system username if the Seller is not found.
- **ApprovedBySecondary** – Display element, returns the secondary part of the Approved By name. The secondary part is displayed only if the name and surname of the Seller are known and contain a system username in brackets.
- **TransactionItems** – Returns a list of all transactions that were considered when calculating the values in the agreement. The transaction items have to be returned as a list of maps where each map reflects one item, the structure is clearly defined and has to map 1-1 with what is in the publishing template.
- **TotalTransactions** – Display element, returns the count of all transactions.

SC_CompensationRecordCalculationFeeder

Architecture Documentation

https://pricefx.atlassian.net/wiki/display/ACC/Architecture+Documentation+(SC)

Logic Description

This is a feeder logic used for fetching and providing items for scheduled compensation calculation executions.

Elements Description

- **SyntaxCheckAbort**
- **EmittingCompensationRecord** – Main element that emits the Compensation Records based on the provided filters.

SC_AdjustmentHeader

Architecture Documentation

https://pricefx.atlassian.net/wiki/display/ACC/Architecture+Documentation+(SC)

Logic Description

This is a header logic for the Adjustment Custom Form object. It is used to read the properties of the object and pass them to the header configurator.

Elements Description

- **Configurator** – Initializes the header configurator in the customFormProcessor and passes appropriate parameters to it.

SC_AdjustmentHeaderConfigurator

Architecture Documentation

https://pricefx.atlassian.net/wiki/display/ACC/Architecture+Documentation+(SC)

Logic Description

This is the main Adjustment object configurator that is used to generate adjustments based on user provided inputs.

Elements Description

- **DatamartName** – Returns the Datamart name stored in in Advanced Configuration.
- **SellerData** – Returns all Sellers (with labels) stored in the system.
- **AdjustmentFormId** – Retrieves the adjustment ID value passed from the header logic.
- **CreatedByInput** – Returns a read only text input that displays the Created By value.
- **CreatedOnInput** – Returns a read only text input that displays the Created On value.
- **TitleInput** – Returns a read only text input that displays the Adjustment title.
- **StatusInput** – Returns a read only text input that displays the current Adjustment status.
- **DescriptionInput** – Returns a text input that allows the user to insert a description for the Adjustment.
- **ProductGroupInput** – Returns a Product Group selection input used to filter out transactions affected by the Adjustment.
- **CustomerGroupInput** – Returns a Customer Group selection input used to filter out transactions affected by the Adjustment.
- **SellerInput** – Returns a Seller selection input used to filter out transactions affected by the Adjustment.
- **PaymentDateInput** – Returns a Payment Date selection input used to filter out transactions affected by the Adjustment.

- **GeneralFilterInput** - Returns a General Filter input used to filter out transactions affected by the Adjustment.
- **TransactionsDataMatrix** - Displays the transaction matrix that shows transactions that will be affected by the adjustment. The display limit is set to show top 10 transactions, the rest is hidden due to performance reasons.
- **AdjustmentTypeInput** - Returns an option input that allows the selection of the Adjustment type that will define how the adjustment will be generated and handled.
- **NewSellerInput** - In case of "Change Data" Adjustment type allows to insert a new Seller value for the selected transactions.
- **NewProductInput** - In case of "Change Data" Adjustment type allows to insert a new Product value for the selected transactions.
- **NewCustomerInput** - In case of "Change Data" Adjustment type allows to insert a new Customer value for the selected transactions.
- **NewDateInput** - In case of "Change Data" Adjustment type allows to insert a new Pricing Date value for the selected transactions.
- **BaselineInputsInfo** - In case of "Change Data" Adjustment type displays some additional information for the user about baseline fields.
- **NewInvoicePriceInput** - In case of "Change Data" Adjustment type allows to insert a new Invoice Price value for the selected transactions.
- **NewQuantityInput** - In case of "Change Data" Adjustment type allows to insert a new Quantity value for the selected transactions.
- **NewGrossMarginInput** - In case of "Change Data" Adjustment type allows to insert a new Gross Margin value for the selected transactions.
- **NewSellersCreditSplit** - In case of "Split Credit" Adjustment type displays an Input Matrix that allows the user to define the Seller credit split. If the split percentage does not equal to 100%, an error message is shown.

SC_AdjustmentWizardExecutor

Architecture Documentation

https://pricefx.atlassian.net/wiki/display/ACC/Architecture+Documentation+(SC)

Logic Description

This is an executor logic for the Adjustment Configurator Wizard, used to create adjustment objects.

Elements Description

- **CreateAdjustmentCFO** -- Generates the adjustment object based on the provided inputs and stores the appropriate values in attributes.
- **SummaryScreen** -- Displays a summary message informing the user that what happened after submission of the Adjustment.

SC_AdjustmentWizardInputs

Architecture Documentation

https://pricefx.atlassian.net/wiki/display/ACC/Architecture+Documentation+(SC)

Logic Description

This is an input logic for the Adjustment Configurator Wizard, used to display inputs for the user.

Elements Description

- **SellerData** -- Returns all Sellers (with labels) stored in the system.
- **TitleInput** -- Generates the text input that allows the user to insert the title of the adjustment.
- **DescriptionInput** - Generates the text input that allows the user to insert the description of the adjustment.
- **ProductGroupInput** -- Generates the Product Group input that allows the user to provide transactions for the Adjustment.
- **CustomerGroupInput** -- Generates the Customer Group input that allows the user to provide transactions for the Adjustment.
- **SellerInput** -- Generates the Seller options selection input that allows the user to provide transactions for the Adjustment.
- **PaymentDateInput** -- Generates the Date input that allows the user to provide transactions for the Adjustment.
- **GeneralFilterInput** -- Generates the General Filter input that allows the user to provide transactions for the Adjustment.

SC_QuotingPluginLib

Architecture Documentation

https://pricefx.atlassian.net/wiki/display/ACC/Architecture+Documentation+(SC)

Logic Description

This is a library that is used for integration with quoting solutions. It allows you to inject the Sales Compensation into the calculation results. The library also supports displaying the results in a pie chart or line score chart.

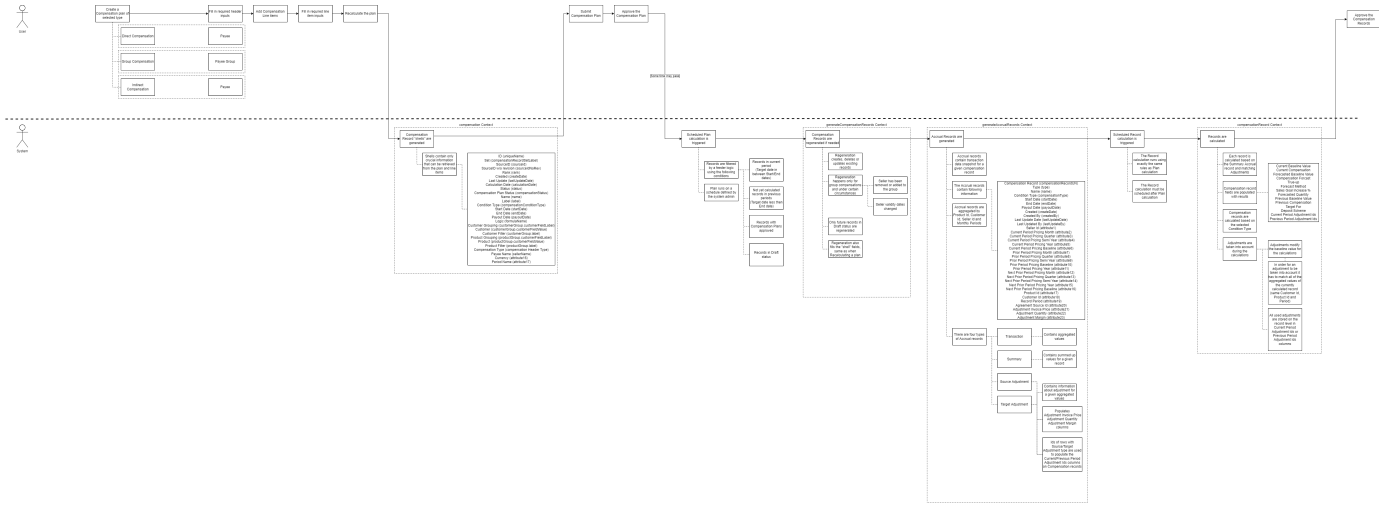
Elements Description

- **ConstConfig** -- Contains definitions of global constant variables that are used throughout the logic. Any configuration-like constants should be stored here to avoid hardcoding and to allow for better maintainability of the solution
- **ConfigManager** - Strictly defined structure that allows management of solution configuration. The benefit is that the whole configuration is contained within one "class", it cannot be modified and is cached.
- **SalesCompensationEstimation** - Stores methods used to calculate the estimated compensation based on items stored in a quote.
- **ChartUtils** - Contains methods used to generate the charts supported by the solution.

Flow Diagrams (SC)

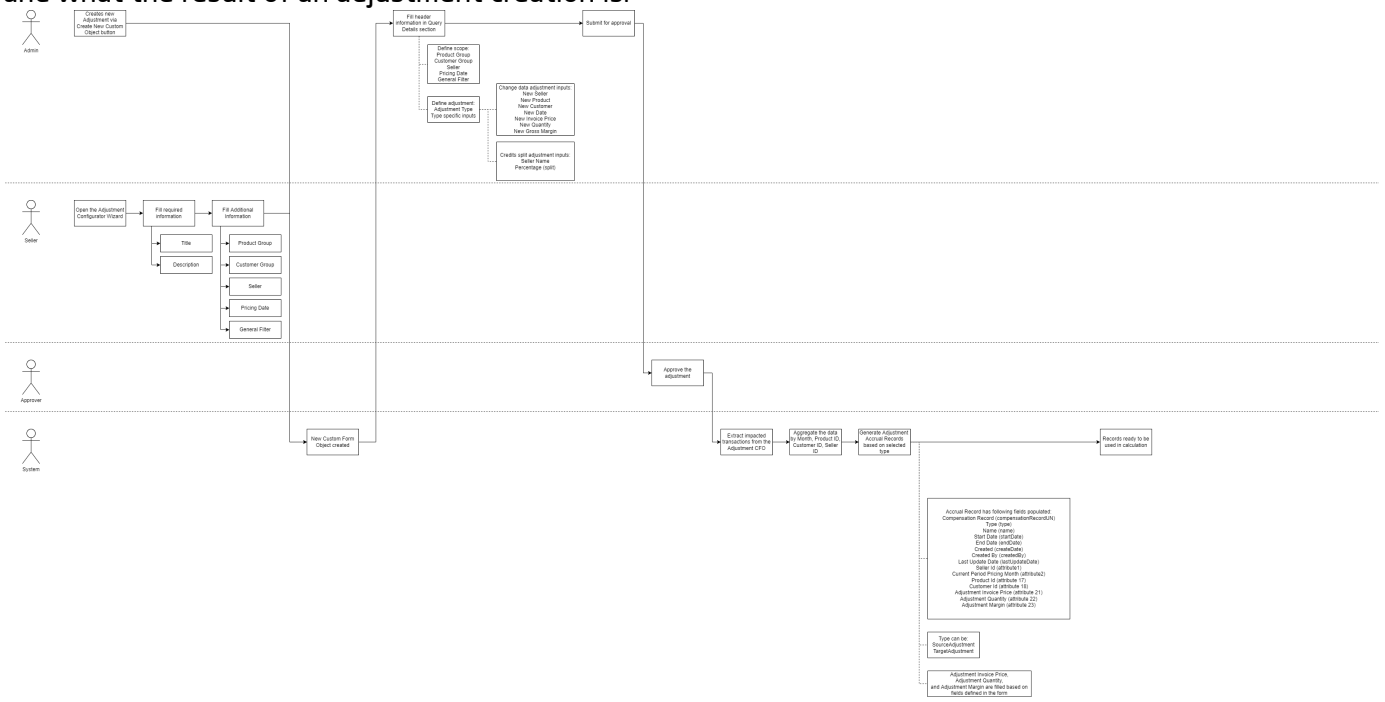
Calculation Flow

The following diagram represents the main calculation flow of the solution, starting from the creation of a plan, through generation of accrual records, adjustments and ending with the calculation results. This is the core of the solution.



Adjustments Flow

The following diagram represents the flow of adjustments, how they are created, what impact they have and what the result of an adjustment creation is.



Manual Upgrade Steps (SC)

In this section, you will find all manual steps which need to be done while upgrading between versions. A description of the whole upgrade procedure is described in the Upgrade (SC) section. In addition, you can also check out the latest release notes.

1.0.0 1.1.0

We do not support upgrading between these versions. 1.1.0 uses the Sales Compensations module instead of the Rebates module, so it needs to be deployed from scratch.

Sales Compensation 1.4.0

⚠ Version 1.4.0 requires Pricefx Hurricane 9.3.0 to work properly.

New Condition Types

New Condition Types have been added with this release:

- Multi Target Bonus
- Stepped Bonus
- Variable Pay Linear Amount
- Variable Pay Multi Target Amount
- Variable Pay Stepped Amount

New Seller Extension

A new `Seller Salaries` Seller Extension has been added. It is used to calculate in the Variable Pay Linear Amount, Variable Pay Multi Target Amount, and Variable Pay Stepped Amount.

Adjustments Limits

Currently, when you submit the Adjustment, with filters on Transactions data returning data exceeding 25 000 rows, you often get timeout. This issue will be resolved once the support for mass addOrUpdate for COARs is added (PFUN-17298).

Upgrade Notes

Due to new Condition Types being added, there are several changes that need to happen during upgrade from 1.3.0 to 1.4.0. The changes are:

Add New Condition Types in `SC_ConditionTypes` Company Parameter

Condition Type Code	Condition Type Name
MultiTargetBonus	Multi Target Bonus
SteppedBonus	Stepped Bonus
VariablePayLinearAmount	Variable Pay Linear Amount
VariablePayMultiTargetAmount	Variable Pay Multi Target Amount
VariablePaySteppedAmount	Variable Pay Stepped Amount

Create New Condition Types

Name	Label	Pricing Logic (formulaName)	Condition Type Name (attribute4)
SC_MultiTargetBonus	Multi Target Bonus	Sales Compensation (SC_Compensation)	Multi Target Bonus
SC_SteppedBonus	Stepped Bonus	Sales Compensation (SC_Compensation)	Stepped Bonus
SC_VariablePayLinearAmount	Variable Pay Linear Amount	Sales Compensation (SC_Compensation)	Variable Pay Linear Amount
SC_VariablePayMultiTargetAmount	Variable Pay Multi Target Amount	Sales Compensation (SC_Compensation)	Variable Pay Multi Target Amount
SC_VariablePaySteppedAmount	Variable Pay Stepped Amount	Sales Compensation (SC_Compensation)	Variable Pay Stepped Amount

Adjust Pricefx Parameter

`SC_AdjustmentWizardTransactionFieldMapping`:

- Table Type: SIMPLE MATRIX
- Value Type: MATRIX MATRIX 2
- key1:
  - Name: adjustmentFieldName
  - Label: Adjustment Field Name
  - Type: String
  - Required
- key2:
  - Name: datamartFieldName
  - Label: Datamart Field Name
  - Type: String
  - Required

Update Data in `Sales Organisation` Seller Extensions

Due to the change made in PFPCS-6246, it is required to update the existing data in the `Sales Organisation` Seller Extension based on the new structure as shown below:

- attribute1 Team
- attribute2 Valid From
- attribute3 Valid To

Add New `Seller Salaries` Seller Extension

- Name: SellerSalaries
- Label: Seller Salaries
- Size: 3
- Allow Search: True
- User Group (Edit): [SC] Administrator
- User Group (View Details): [SC] Administrator

Details:

- Business Key: Seller ID
- attribute1:
  - Name: FixedSalary
  - Label: Fixed Salary (Annual)
  - Type: Real
  - Format Type: Money
  - Required

Stories

PFPCS-5980 Multi Target Bonus

PFPCS-5982 Stepped Bonus

PFPCS-6275 Variable Pay Linear Amount

PFPCS-6277 Variable Pay Multi Target Amount

PFPCS-6276 Variable Pay Stepped Amount

PFPCS-5901 Group Compensation - show warnings when COR generation is skipped because of validity dates

PFPCS-5938 Line Item Compensation and Forecast summary on Header

PFPCS-6196 Address the TODO in Admin dashboard related to PFUN-16836

PFPCS-6197 Dashboard Improvements - Transaction Dashboard

PFPCS-6224 Address changes made in User Roles

PFPCS-6255 Dashboards - User permissions to be driven by User Groups instead of Business Roles

Improvements

PFPCS-6236 Add Adjustment Wizard Transaction Field Mapping validation

Bug Fixes

PFPCS-6030 Adjustments - Clicking submit before "applying" input causes an error and breaks data

PFPCS-6296 Adjustment does not generate accrual records

PFPCS-6307 Input validation still shows when all Target Rate (%) inputs were filled

PFPCS-6310 Fixed Salary should be changed to Annual Fixed Salary

PFPCS-6229 Group: Previous baseline value on CO is displayed incorrectly because of validFrom in Sales Org SX

PFPCS-6266 Error thrown when leaving Sales Goal Increase % input empty on header

PFPCS-6102 SC Transaction Dashboard - Only Payee who participate in COR should be displayed in Payee selection input

PFPCS-6262 PayoutTable : ERROR(@0): No signature of method: parsedscript_PayoutTable_fZKNIjimVO.createPayoutTable() when opening Admin Dashboard and Payouts and Plans Dashboard

PFPCS-6280 SC Compensation Dashboard - error CompensationInput is displaying

PFPCS-6281 SC Transaction Dashboard - Error CompensationRecordInput is displayed when applying Seller

PFPCS-6282 SC Transaction Dashboard - Error CompensationPlanStatistics is displayed when applying only Compensation Plan Type and Compensation Plan

PFPCS-6283 SC Transaction Dashboard - Incorrect input field names and default date

PFPCS-6284 User permissions priority on SC Dashboards

PFPCS-6285 Multi Target Bonus: error thrown when adding and recalculating this condition type

PFPCS-6289 SC Compensation Dashboard - No data is displayed if Compensation Plan is empty

PFPCS-6290 SC Transaction Dashboard - Compensation Records portlet shows all data and was not affected by Seller Id if only Seller Id is applied

PFPCS-6292 Multi Target Bonus: Missing validation "Target Incentive should be less than Quota"

PFPCS-6297 SC Transaction Dashboard - Incorrect configuration for Date input

PFPCS-6298 Variable Pay: Change Condition Type name

PFPCS-6302 Variable Pay Linear Amount: Current salary is not divided by the periods defined in the calculation

PFPCS-6303 Multi Target Bonus: Incorrect compensation value

PFPCS-6309 Stepped Bonus: Incorrect compensation value

PFPCS-6312 Validation message shows improperly when having no configuration in SC Adjustment Wizard Transaction Field Mapping PP

PFPCS-6314 Multi Target Variable Pay: Change Condition Type name

PFPCS-6315 Variable Stepped Amount: Incorrect compensation value

PFPCS-6318 Transactions Dashboard - Admin access User Group should be removed

PFPCS-6246 Attribute 1 (old managerOf field) is still visible in Sales Org SX

Tasks

PFPCS-6142 Do the performance investigation of SC on dev partition to check why jobs are working so slowly

PFPCS-6300 Add unit test for each of the Condition Types compensation calculations