Sample Content for PCM to PaPM Activity Based Costing
SAP Profitability and Performance Management
Document Version: 3.0 SP15 – 2021-09-21

PUBLIC

Sample Content for PCM to PaPM Activity Based Costing

Processes and Functions supporting Sample Business Scenario

Typographic Conventions

Type Style	Description
Example	Words or characters quoted from the screen. These include field names, screen titles, pushbuttons labels, menu names, menu paths, and menu options. Textual cross-references to other documents.
Example	Emphasized words or expressions.
EXAMPLE	Technical names of system objects. These include report names, program names, transaction codes, table names, and key concepts of a programming language when they are surrounded by body text, for example, SELECT and INCLUDE.
`Example`	Output on the screen. This includes file and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools.
`Example`	Exact user entry. These are words or characters that you enter in the system exactly as they appear in the documentation.
`<Example>`	Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.
`EXAMPLE`	Keys on the keyboard, for example, `F2` or `ENTER`.

Contents

Table of Figures

1 Introduction and Basics

1.1 About this Guide

This guide provides information about the sample content PCM to PaPM Activity Based Costing, which can be installed on top of SAP Profitability and Performance Management. This sample content describes a project accelerator, ideas, and best practices for modeling an end-to-end profitability and cost calculation model that is feasible for actuals, planning, forecasting and simulations. It also contains references to further documentation that you should read before performing these tasks.

The structure of this document is organized around the following topic:

Business Example
This part of the guide covers the main features of the sample content. It describes the information model and calculation model.

Target Audience:

- Business experts
- Solution consultants
- Presales teams

Considerations

It is essential to be accustomed with the content of the corresponding guides and documents related to this topic before beginning with this example. For more information about the available guides and documents, as well as integration with other systems, roles, configuration information, users and authorization concept, see section Related Documentation.

1.2 Constraints

This guide does not provide information about the installation of the sample content. For more information about this, see section Related Documentation.

1.3 Related Documentation

The following table lists related documents.

Topic	Guide/Tool/Title	Links
• Installation and planning of your system landscape • Activities to keep the system running • Information about how to ensure the required security for your SAP landscape	Administration Guide	https://help.sap.com/viewer/a8068ea2f9ba403aa6d8bda3abcdbecd/latest/en-US
• Sample content for PCM to PaPM Activity Based Costing	Sample content for PCM to PaPM Activity Based Costing	https://launchpad.support.sap.com/#/notes/2933822 SAP Note 2933822
• Operation of SAP NetWeaver	Technical Operations Manual	https://help.sap.com/viewer/p/SAP_NETWEAVER_750
• Application Help	Detailed Application help for SAP Profitability and Performance Management	https://help.sap.com/viewer/56471df1959f4cfd9e3bf7a6d2d5be42/latest/en-US
• SAP HANA Administration Guide	Administration guide for SAP HANA; supported SDA databases	https://help.sap.com/viewer/product/SAP_HANA_PLATFORM/
• SAP Notes		https://launchpad.support.sap.com/

1.4 Glossary

ABC	Activity-based costing
BI	Business intelligence
BW	Business warehouse
PaPM	SAP Profitability and Performance Management

2 Business Example

The market is at the dawn of the next big technology change where everything is connected and software is embedded in people's lives. This technology change is bringing new opportunities and new threats. Cycle time for innovation is 5–10 times faster, and enterprises can no longer compete unless complexity is reduced. Business efficiency is ahead of the market and product and service profitability are constantly tracked and optimized.

That is why digital performance management will be the game-changer for companies who want to be successful in the digital economy. A digital performance management solution for 21st century business needs to measure and manage enterprise efficiency and drive product and service profitability in real time.

Built on SAP HANA, SAP Profitability and Performance Management is a next generation digital performance management solution that provides breakthrough real-time business data aggregation capabilities for SAP and non-SAP systems, a high-speed finance and risk calculation engine and comprehensive simulation and scenario management.

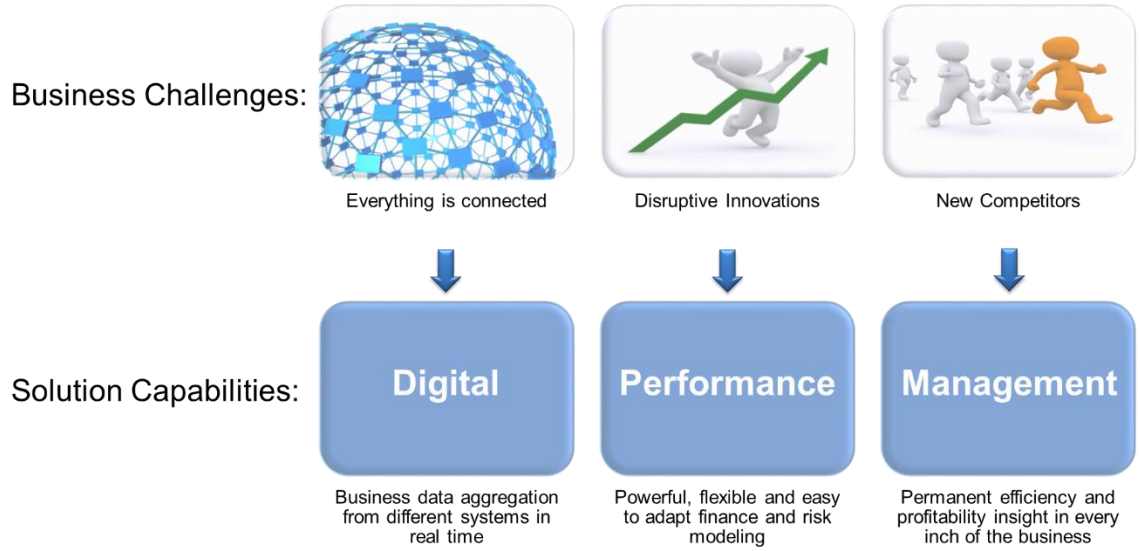

Figure 1: Digital Imperative

2.1 Scope and Business Definition

This sample content covers the sample end-to-end profitability and cost process of an enterprise, comprising certain aspects of data integration, data input, processing (including calculations and allocations following an ABC (activity-based costing) approach as well as reporting.

This allows business users to manage and analyze enterprise profitability and cost in one central solution.

The following screenshot shows the function hierarchy of the sample content and the process template.

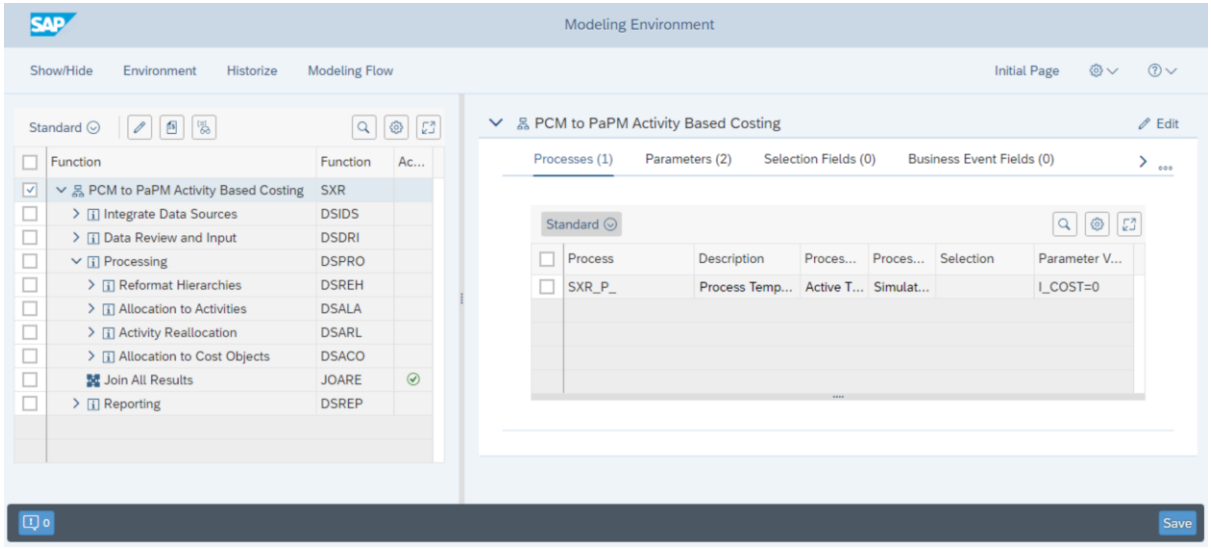

Figure 2: Sample Content Information and Calculation Model

You can also find the information contained in the following chapters on the respective *Documentation* tabs for the model.

3 PCM to PaPM Activity Based Costing in Detail

PCM to PaPM Activity Based Costing is based on Profitability and Cost Management methodology (PCM). It incorporates an allocation process that is based on PCM rules. This sample content is made to demonstrate all possibilities of SAP Profitability and Performance Management to replicate calculations that are performed in PCM. Specific flat hierarchies are necessary in order to create model on the most granular level and it is the first step in model building. Profitability and Cost Management process (PCM) is based on Activity Based Costing and allocation process is consisted of three phases:

- Allocation to Activities
- Activity Reallocation - Activity Reassignment
- Allocation of Activity costs to Cost Objects

This sample content for PCM to PaPM Activity Based Costing (Environment ID=SXR, Version=3) covers an end-to-end example of an activity-based costing model. It incorporates a predefined process template with activities to run the model in production as well as for what-if simulation purposes.

This sample content comes with the SAP Profitability and Performance Management software installation.

It consists of one *Calculation Unit* function that is structured underneath by Description functions in the following sections:

1. Integrate Data Sources
   Defines manually loaded data used in the subsequent functions.
   - *Parent - Child Data*
   - *Line Item Values*
   - *Line Item Activity Rules*
   - *Line Item Resource Drivers*
   - *Activity Reallocation Data*
   - *Resource Driver Split*
   - *Cost Object Assignment Data*
   - *Cost Object Drivers*
   - *Activity Driver Values*

2. Data Review and Input
   Provides specific input options for the following:
   - *Review and Update Parent - Child Data*
   - *Review and Update Line Item Values*

- *Review and Update Line Item Activity Rules*
- *Review and Update Line Item Resource Drivers*
- *Review and Update Activity Reallocation Data*
- *Review and Update Resource Driver Split*
- *Review and Update Cost Object Assignment Data*
- *Review and Update Cost Object Drivers*
- *Review and Update Activity Driver Values*

3. Processing
   Here the core of the calculation model is defined to get to a complete multidimensional P&L at granular level.
   - *Reformat Hierarchies*
     - *Execute hierarchy format change*
     - *Format change logic*
       - *Flat Hierarchies Table*
       - *Derive first 2 hierarchy levels*
       - *Initialize FHT*
       - *Hierarchies Iteration View*
       - *Iteration Logic*
         - *Get PC Data*
         - *Derive fields in FHT*
         - *Write to FHT*
       - *Responsibility Center hierarchy reformatted*
       - *Activity hierarchy reformatted*
       - *Cost Object 1 hierarchy reformatted*
       - *Cost Object 2 hierarchy reformatted*
       - *Cost Object 3 hierarchy reformatted*
       - *Cost Object 4 hierarchy reformatted*
       - *Cost Object 5 hierarchy reformatted*
   - *Allocation to Activities*
     - *Apply LIDACT Rules*
     - *Apply LIRD Rules*
     - *Perform allocation to activities*
     - *Collect Activity Results*
   - *Activity Reallocation*
     - *Prepare receiver at lowest granularity*
     - *Get Drivers*
     - *Activity Reallocation*

- *Collect Activity Results*

- *Allocation to Cost Objects*
  - *Sender Preparation - Apply COD rules*
  - *Receiver preparation - Apply COA rules*
  - *CO1 Allocation*
    - *CO1 Receiver Data*
    - *CO1 Level 3007 - RC Unassigned*
      - *CO1 get drivers 3007*
      - *CO1 allocate level 3007*
    - *CO1 Level 0 – All RC's*
      - *CO1 get drivers 0*
      - *CO1 allocate level 0*
    - *CO1 Level 2 - Current RC*
      - *CO1 get drivers 2*
      - *CO1 allocate level 2*
    - *CO1 Level 1 - Parent RC*
      - *CO1 Get RC parent value*
      - *CO1 Get siblings*
      - *CO1 get drivers level 1*
      - *CO1 allocate level 1*
    - *CO1 execute allocation*
  - *CO2 Allocation*
    - *CO2 get receiver data*
    - *CO2 Level 3007 - RC Unassigned*
      - *CO2 Get Drivers 3007*
      - *CO2 allocate level 3007*
    - *CO2 Level 0 - All RC's*
      - *CO2 get drivers 0*
      - *CO2 allocate level 0*
    - *CO2 Level 2 - Current RC*
      - *CO2 get drivers 2*
      - *CO2 allocate level 2*
    - *CO2 Level 1 - Parent RC*
      - *CO2 Get RC parent value*

- *CO2 Get siblings*
- *CO2 get drivers level 1*
- *CO2 allocate level 1*

CO2 execute allocation

CO3 Allocation

CO3 get receiver data

CO3 Level 3007 - RC Unassigned

- *CO3 Get Drivers 3007*
- *CO3 allocate level 3007*

CO3 Level 0 – All RCs

- *CO3 get drivers 0*
- *CO3 allocate level 0*

CO3 Level 2 - Current RC

- *CO3 get drivers 2*
- *CO3 allocate level 2*

CO3 Level 1 - Parent RC

- *CO3 Get RC parent value*
- *CO3 Get siblings*
- *CO3 get drivers level 1*
- *CO3 allocate level 1*

CO3 execute allocation

– *Join All Results*

4. Reporting, which provides review reports for different dimensions: *Activity, Cost Object Dimensions, Responsibility Center.*

The *PCM to PaPM Activity Based Costing* calculation unit defines the process template **SXR_P_ - Process Template** with the following activities:

- ***Update Input Attributes:***
  - *Review and Update Parent - Child Data*, where the user can check and change dimensions and hierarchies in the process
  - *Review and Update Line Item Activity Rules,* where the user can get insight into activities and apply changes if needed
  - *Review and Update Line Item Resource Drivers,* where the user can get insight into resource drivers and apply changes if necessary
  - *Review and Update Activity Reallocation Data,* where it is possible to check rules that are applied in re-allocation process

- *Review and Update Cost Object Allocation,* where it is possible to check and update rules that are applied in cost object allocation process
- *Review and Update Cost Object Drivers,* where it is possible to check and update drivers that are applied in cost object allocation process
- *Review and Update Line Item Values*, where the user can apply changes in values
- *Review and Update Resource Driver Split*, where changes can be applied to the allocation logic if needed
- *Review and Update Activity Driver Values*, where changes can be applied to the allocation logic if needed

- ***Execution***

- ***Reporting:***
  - *Activity Results*, which provides a predefined activity result report
  - *Activity Results [Side-by-side Simulation]*, which provides a predefined activity result report
  - *Activity Reallocation Results*, which provides results of activity reassignment
  - *Cost Object 1 Results*, which provides a predefined result for Cost Object 1
  - *Cost Object 2 Results*, which provides a predefined result for Cost Object 2
  - *Cost Object 3 Results*, which provides a predefined result for Cost Object 3
  - *Activity and Cost Object Results,* which shows results of cost objects and contribution of each activity.

Note

All execution activities are configured without dependency, so that they can be executed in parallel for demo purposes. In implementation projects dependent activities can be defined instead if needed.

For execution activities no performer and reviewer teams are assigned, that means every execution user can run the activities. In implementation projects teams need to be assigned to restrict the authorizations to relevant users.

For the used environment fields, no characteristic based authorization is defined, that means every execution user can read all data and possibly apply data changes. In implementation projects authorizations can be defined on characteristic value level by IT Administrators in line with the general SAP authorization and security management, so that for example in the “Review Financial Statement Item” report the company data for “Sunshine New York” is only readable to selected users. Another example would be, that for "Update Plan and Forecast Data" team members can only input data for cost centers, for which they are responsible.

3.1 Integrate Data Sources

In this section, you can define the required data sources.

Note

This sample content does not work with specific customer data and cannot presume any customer-specific system landscape, application or interface.

The integration of data sources uses therefore functions of type Model Table to make test data available in an implementation project to connect to the real and concrete customer data sources and targets.

The complete information model is kept lean to ease the adaption in an implementation project.

This section defines manually loaded data used in the subsequent functions:

- *Parent - Child Data*
- *Line Item Values*
- *Line Item Activity Rules*
- *Line Item Resource Drivers*
- *Activity Reallocation Data*
- *Resource Driver Split*
- *Cost Object Assignment Data*
- *Cost Object Drivers*
- *Activity Driver Values*

3.1.1 Parent-Child Data

Parent-child model table contains data that defines all members of the hierarchies in the model for all dimensions. This model table is used as input function in reforming hierarchies.

Fields:

1. *Dimension:*

   It is used to represent all available dimensions in one business process

2. *Parent:*

   This field describes dimension member in hierarchy, usually higher members of hierarchy

3. *Child:*

   This field describes dimension member in hierarchy, usually lower members of hierarchy

4. *Attribute Flag:*

   This field specifies structure of dimension which is under one attribution flag

5. *Status:*

It indicates if the parent-child relation is used in the model: 1 - relation is active used; 0 - relation is not used in the model.

3.1.2 Line Item Values

Line item Values Model table contains amount of costs that should be allocated in model. This Model table is input for generating sender data.

It defines the following fields:

1. *Version:*

   This field describes which version of data is used (e.g. Actual, Forecast…)

2. *Period:*

   The field represents the period to which costs and data are related

3. *Responsibility Center:*

   This field describes hierarchical structure of business units, departments, or cost centers

4. *Line Item:*

   The field describes basic level of costs that are usually taken from General Ledger

5. *Currency:*

   This field describes currency as unit of measure

6. *Amount:*

   This filed contains all values of costs that are used in processing

3.1.3 Line Item Activity Rules

This Model Table specifies all rules for direct allocation, since this model is based on PCM and data driven approach.

It defines the following fields:

1. *Version:*

   This field describes which version of data is used (e.g. Actual, Forecast…)

2. *Period:*

   The field represents the period to which costs and data are related

3. *Responsibility Center:*

   This field describes hierarchical structure of business units, departments, or cost centers

4. *Line Item:*

The field describes basic level of costs that are usually taken from General Ledger

5. *Activity:*

   The Activity field represents all activities that are related to business process

6. *Status:*

   It indicates if the corresponding entry is used in the model: 1 - entry is active used; 0 - entry is not used in the model.

3.1.4 Line Item Resource Drivers

This Model Table function specifies drivers that are used as distribution bases in allocation of costs to activities.

It defines the following fields:

1. *Version:*

   This field describes which version of data is used (e.g. Actual, Forecast…)

2. *Period:*

   The field represents the period to which costs and data are related

3. *Responsibility Center:*

   This field describes hierarchical structure of business units, departments, or cost centers

4. *Line Item:*

   The field describes basic level of costs that are usually taken from General Ledger

5. *Resource Driver:*

   Represents different measures in business that are used as distribution base for different allocations

6. *Status:*

   It indicates if the corresponding entry is used in the model: 1 - entry is active used; 0 - entry is not used in the model.

3.1.5 Activity Reallocation Data

Activity Reallocation Data is a model table that defines how the activity reallocation is performed in model. Sender activity and Sender Responsibility Center are defined so that allows to track iterative allocation in processing part.

It defines the following fields:

1. *Version:*

   This field describes which version of data is used (e.g. Actual, Forecast…)

2. *Period:*

   The field represents the period to which costs and data are related

3. *Sender Responsibility Center:*

   This field appears to describe responsibility center from which activities are reassigned

4. *Sender Activity:*

   This field is used to describe activity that should be used in reassigning process

5. *Resource Driver 1:*

   Driver field is used to describe resource driver that is used

6. *Resource Driver 2:*

   Driver field is used to describe resource driver that is used

7. *Responsibility Center:*

   This field describes hierarchical structure of business units, departments, or cost centers

8. *Activity:*

   The Activity field represents all activities that are related to business process

9. *Status:*

   It indicates if the corresponding entry is used in the model: 1 - entry is active used; 0 - entry is not used in the model.

3.1.6 Resource Driver Split

This Model Table function contains data related to values of drivers that are used in cost allocation. It is used in Activity Allocation and Activity Reallocation parts, since this Model Table contains activity related to those drivers.

It defines the following fields:

1. *Version:*

   This field describes which version of data is used (e.g. Actual, Forecast…)

2. *Period*:

   The field represents the period to which costs and data are related

3. *Responsibility Center:*

   This field describes hierarchical structure of business units, departments, or cost centers

4. *Resource Driver:*

   Represents different measures in business that are used as distribution base for different allocations

5. *Activity:*

   The Activity field represents all activities that are related to business process

6. *Driver Amount:*

   This fields contains values of drivers that are used as distribution

3.1.7 Cost Object Assignment Data

This Model Table function defines all target cost objects for allocation of activities to cost objects. Besides, it contains rules that are applied in cost object allocation process.

It defines the following fields:

1. *Version:*

   This field describes which version of data is used (e.g. Actual, Forecast…)

2. *Period:*

   The field represents the period to which costs and data are related

3. *Responsibility Center:*

   This field describes hierarchical structure of business units, departments, or cost centers

4. *Activity:*

   The Activity field represents all activities that are related to business process

5. *Cost Object Generic:*

   This specifies different categories of output in one organization

6. *Dimension:*

   Used to represent all available dimensions in one business process

7. *Status:*

   It indicates if the corresponding entry is used in the model: 1 - entry is active used; 0 - entry is not used in the model.

3.1.8 Cost Object Drivers

Cost Object Drivers Model Table function defines the drivers that are used to allocate *Activities* to *Cost Objects*. In the list below, user can notice that every cost object driver has its driver level. Based on driver level, allocation on cost object is performed.

It defines the following fields:

1. *Version:*

   This field describes which version of data is used (e.g. Actual, Forecast…)

2. *Period:*

   The field represents the period to which costs and data are related

3. *Responsibility Center:*

   This field describes hierarchical structure of business units, departments, or cost centers

4. *Activity:*

   The Activity field represents all activities that are related to business process

5. *CO1 Driver:*

   This field describes drivers related to Cost Object 1

6. *CO2 Driver*

   This field describes drivers related to Cost Object 2

7. *CO3 Driver*

   This field describes drivers related to Cost Object 3

8. *CO4 Driver*

   This field describes drivers related to Cost Object 4

9. *CO5 Driver*

   This field describes drivers related to Cost Object 5

10. *CO1 Driver Level*

    This field describes levels of drivers that are used in allocation process

11. *CO2 Driver Level*

    This field describes levels of drivers that are used in allocation process

12. *CO3 Driver Level*

    This field describes levels of drivers that are used in allocation process

13. *CO4 Driver Level*

    This field describes levels of drivers that are used in allocation process

14. *CO5 Driver Level*

    This field describes levels of drivers that are used in allocation process

15. *Status:*

    It indicates if the corresponding entry is used in the model: 1 - entry is active used; 0 - entry is not used in the model.

3.1.9 Activity Driver Values

This Model Table function represents all driver values that are used to allocate activities to cost objects.

It defines the following fields:

1. *Version:*

   This field describes which version of data is used (e.g. Actual, Forecast…)

2. *Period:*

   The field represents the period to which costs and data are related

3. *Responsibility Center:*

   This field describes hierarchical structure of business units, departments, or cost centers

4. *Cost Object Driver:*

   This field describes all drivers that are used to allocate costs on different cost objects

5. *Driver Amount:*

   This field contains values of cost object values that are used as distribution base in allocations

6. *Cost Object 1:*

   This field describes item that allows identifying a cost that should be reported, for example a product, customer, distribution channel, or any combination of some of these.

7. *Cost Object 2:*

   This field describes item that allows identifying a cost that should be reported, for example a product, customer, distribution channel, or any combination of some of these.

8. *Cost Object 3:*

   This field describes item that allows identifying a cost that should be reported, for example a product, customer, distribution channel, or any combination of some of these.

9. *Cost Object 4:*

   This field describes item that allows identifying a cost that should be reported, for example a product, customer, distribution channel, or any combination of some of these.

10. *Cost Object 5:*

    This field describes item that allows identifying a cost that should be reported, for example a product, customer, distribution channel, or combination of some of these.

3.2 Data Review and Input

In this section additional Query functions for comfortable review and maintenance of data by execution users are defined.

In the process, this review and maintenance activities happen, before the Profitability and Cost Management Calculation is executed.

- *Review and Update Parent - Child Data*, where the user can check and make changes in dimensions and hierarchies in the process
- *Review and Update Line Item Values*, where the user can apply changes in values
- *Review and Update Line Item Activity Rules,* where the user can get insight into activities and make changes if needed
- *Review and Update Line Item Resource Drivers,* where the user can get insight into resource drivers and make changes if needed
- *Review and Update Activity Reallocation Data,* where is possible to check rules or add new one that are applied in re-allocation process
- *Review and Update Resource Driver Split*, where changes can be applied to the allocation logic if needed
- *Review and Update Cost Object Allocation,* where is possible to check and change rules that are applied in cost object allocation process
- *Review and Update Cost Object Drivers,* where is possible to check and change drivers that are applied in cost object allocation process
- *Review and Update Activity Driver Values*, where changes can be applied to the allocation logic if needed

Note

Edit access queries offer sometimes only a subset of the test data for editing to show-case that functionality.

3.2.1 Review and Update Parent - Child Data

This Query function is used to provide read and edit data access to parent-child data, which serves as an input for the PCM to PaPM calculation.

See Parent - Child Data function for details about the fields.

3.2.2 Review and Update Line Item Values

This Query function is used to provide read and edit data access to Line Item Values, which serves as an input for the PCM to PaPM calculation.

See Line Item Values function for details about the fields.

3.2.3 Review and Update Line Item Activity Rules

This Query function is used to provide read and edit data access to Line Item Activity Rules, which serves as an input for the PCM to PaPM calculation.

See Line Item Activity Rules for details about the fields.

3.2.4 Review and Update Line Item Resource Drivers

This Query function is used to provide read and edit data access to Line Item Resource Drivers, which serves as an input for the PCM to PaPM calculation.

See Line Item Resource Drivers function for details about the fields.

3.2.5 Review and Update Activity Reallocation Data

This Query function is used to provide read and edit data access to Line Item Resource Drivers, which serves as an input for the PCM to PaPM calculation.

See Activity Reallocation Data function for details about the fields.

3.2.6 Review and Update Resource Driver Split

This Query function is used to provide read and edit data access to Resource Driver Split, which serves as an input for the PCM to PaPM calculation.

See Resource Driver Split function for details about the fields.

3.2.7 Review and Update Cost Object Assignment Data

This Query function is used to provide read and edit data access to Cost Object Allocation Data, which serves as an input for the PCM to PaPM calculation.

See Cost Object Assignment Data function for details about the fields.

3.2.8 Review Cost Object Drivers

This Query function is used to provide read and edit data access to Cost Object Drivers, which serves as an input for the PCM to PaPM calculation.

See Cost Object Drivers function for details about the fields.

3.2.9 Review and Update Activity Driver Values

This Query function is used to provide read and edit data access to Activity Driver Values, which serves as an input for the PCM to PaPM calculation.

See Activity Driver Values function for details about the fields

3.3 Processing

In this section the core functions of the model to perform cost allocation through different dimensions.

In the process this execution happens, after the input data is reviewed and updated.

It comprises the following functions:

1. *Execute hierarchy format change,* which is Join function used to trigger writer for initial data to be written in Flat hierarchies table and View function that contain Iterations
2. *Flat Hierarchies Table,* Model Table function used to store result of iterations and initial data set
3. *Derive first 2 hierarchy levels,* Derivation function is used derive first two levels of hierarchy using parent-child relationship from that model table.
4. *Initialize FHT*, writer functions are used to write initial data set that is used later in processing, especially as input in creating iteration logic
5. *Hierarchies Iteration View*, the View function does iterations and therefore is crucial in iterative process of forming hierarchy levels
6. *Get PC Data,* Join function that is used to prepare data for creating calculation logic for iteration process.
7. *Derive fields in FHT,* Derivation function that is used as calculation logic in iteration process to generate all hierarchies using many rules.
8. *Write to FHT,* writer function is used as input function in *Hierarchies iteration view* and by that indirectly write the result in *Flat hierarchies table*.
9. *Responsibility Center hierarchy reformatted,* the View function that is used after iteration process so specific dimensions could be extracted from *Execute hierarchy format change*,

in this case Responsibility centers, and used in processing depending on different allocation levels.

10. *Activity hierarchy reformatted,* the View function is used in after iteration process so specific dimensions could be extracted from *Execute hierarchy format change*, in this case Activity hierarchies, and used in processing depending on different allocation levels.
11. *Cost Object 1 hierarchy reformatted,* the View function is used in after iteration process so specific dimensions could be extracted from *Execute hierarchy format change*, in this case Cost Object 1 hierarchy reformatted, and used in processing depending on different allocation levels
12. *Cost Object 2 hierarchy reformatted,* the View function that is used in after iteration process so specific dimensions could be extracted from iteration results, in this case Cost Object 2 hierarchy reformatted, and used in processing depending on different allocation levels.
13. *Cost Object 3 hierarchy reformatted,* the View function that is used in after iteration process so specific dimensions could be extracted from *Execute hierarchy format change*, in this case Cost Object 3 hierarchy reformatted, and used in processing depending on different allocation levels.
14. *Cost Object 4 hierarchy reformatted,* the View function that is used in after iteration process so specific dimensions could be extracted from *Execute hierarchy format change*, in this case Cost Object 4 hierarchy reformatted, and used in processing depending on different allocation levels.
15. *Cost Object 5 hierarchy reformatted,* the View function that is used in after iteration process so specific dimensions could be extracted from *Execute hierarchy format change*, in this case Cost Object 5 hierarchy reformatted, and used in processing depending on different allocation levels.
16. *Apply LIDACT Rules*, the Join function that is used to combine *Line Item Values* and *Line Items Activity Rules* in order to get suitable sender data for next step in allocation to activities process.
17. *Apply LIRD Rules,* the Join function that uses *Apply LIDACT Rules* and *Line Item Resource Drivers* as inputs to consider all possible combinations in order to generate resource drivers since the combination of VERSION, PERIOD, RESPONSIBILITY CENTER and LINE ITEM determine the RESOURCE DRIVER.
18. *Perform allocation to activities*, the Allocation function that is used to perform allocation of line item values to activities, since PCM concept considers allocation of costs cost to activities as first part of allocation process.
19. *Collect Activity Results,* the Join function is used to collect all results from allocation and non-allocated costs. In following allocation segment, this function is used to generate sender data.
20. *Prepare receiver at lowest granularity,* the Join function that is used to prepare receiver at lowest granularity since the drivers are usually attached at lowest granularity.
21. *Get Drivers,* the Join function that is used to generate drivers that are used in reallocation of activities.

22. *Activity Reallocation,* the Allocation function is used to perform reassignment of activities between each other. There are also activities that are non-reassign, and those will be allocated in future steps.

23. *Collect Activity Results,* the Join function that uses as inputs *Activity Reallocation* and *Collect Activity Results* from previous allocation in order to prepare data for further allocation steps.

24. *Sender Preparation - Apply COD Rules,* the Join function that is used to determine sender data for allocation of costs on cost objects in combination with cost object drivers.

25. *Receiver preparation - Apply COA rules,* the Join function that is used to determine receiver data for allocation of costs on cost objects in combination with cost object allocation data.

26. *CO1 Receiver Data,* the Join function is used to determine receiver data for Cost Object 1 using reformatted hierarchies in order to get receiver depending on level of cost object.

27. *CO1 Get Drivers 3007,* the Join function is used to extract receiver data from CO1 Receiver Data, which are specific for 3007 level of cost object.

28. *CO1 allocate level 3007,* the Allocation function is used to perform allocation costs to cost object 1, level 3007, based on drivers generated in Join function.

29. *CO1 get drivers 0,* the Join function is used to extract receiver data from CO1 Receiver Data, that are specific to 0 level of cost object.

30. *CO1 allocate level 0,* the Allocation function is used to perform allocation costs to cost object 1, level 0, based on drivers generated in Join function.

31. *CO1 get drivers 2,* the Join function is used to extract receiver data from CO1 Receiver Data, that are specific to 2 level of cost object.

32. *CO1 allocate level 2,* the Allocation function is used to perform allocation costs to cost object 1, level 2, based on drivers generated in Join function.

33. *CO1 Get RC parent value,* the Join function is used to get parent values from *Parent-Child data* and CO1 Receiver data, in order to create all sibling values in next step.

34. *CO1 Get siblings,* the Join function is used to get parent values from *Parent-Child data* and *CO1 Get RC parent data*, in order to create all siblings for future generation of allocation drivers.

35. *CO1 get drivers level 1,* the Join function is used to generate drivers for allocation from CO1 Receiver Data, that are specific to level 1 of cost object.

36. *CO1 allocate level 1*, the Allocation function is used to perform allocation of costs to cost object 1, level 1, based on drivers generated in *CO1 get drivers level 1* Join function.

37. *CO1 execute allocation,* the Join function that is used to collect all allocation results from all levels per Cost Object 1 into one result. Those values are used in future processing.

38. *CO2 Receiver Data,* the Join function is used to determine receiver data for Cost Object 2 using reformatted hierarchies in order to get receiver depending on level of cost object.

39. *CO2 Get Drivers 3007,* the Join function is used to extract receiver data from CO2 Receiver Data, that are specific for 3007 level of cost object.

40. *CO2 allocate level 3007,* the Allocation function is used to perform allocation costs to cost object 2, level 3007, based on drivers generated in Join function.

41. *CO2 get drivers 0,* the Join function is used to extract receiver data from CO2 Receiver Data, that are specific to 0 level of cost object.
42. *CO2 allocate level 0,* the Allocation function is used to perform allocation of costs to cost object 2, level 0, based on drivers generated in Join function.
43. *CO2 get drivers 2,* the Join function is used to extract receiver data from CO2 Receiver Data, that are specific to 2 level of cost object.
44. *CO2 allocate level 2,* the Allocation function is used to perform allocation costs to cost object 2, level 2, based on drivers generated in Join function.
45. *CO2 Get RC parent value,* the Join function is used to get parent values from *Parent-Child data* and CO2 Receiver data, in order to create all sibling values in next step.
46. *CO2 Get siblings,* the Join function is used to get parent values from *Parent-Child data* and *CO2 Get RC parent data*, in order to create all siblings for future generation of allocation drivers.
47. *CO2 get drivers level 1,* the Join function is used to generate drivers for allocation from CO2 Receiver Data, that are specific to level 1 of cost object.
48. *CO2 allocate level 1*, the Allocation function is used to perform allocation of costs to cost object 2, level 1, based on drivers generated in *CO2 get drivers level 1* Join function.
49. *CO2 execute allocation,* the Join function that is used to collect all allocation results from all levels per Cost Object 2 into one result.
50. *CO3 Receiver Data,* the Join function is used to determine receiver data for Cost Object 3 using reformatted hierarchies in order to get receiver depending on level of cost object.
51. *CO3 Get Drivers 3007,* the Join function is used to extract receiver data from CO3 Receiver Data, that are specific for 3007 level of cost object.
52. *CO3 allocate level 3007,* the Allocation function is used to perform allocation costs to Cost Object 3, level 3007, based on drivers generated in Join function.
53. *CO3 get drivers 0,* the Join function is used to extract receiver data from CO3 Receiver Data, that are specific to 0 level of cost object.
54. *CO3 allocate level 0,* the Allocation function is used to perform allocation of costs to Cost Object 3, level 0, based on drivers generated in Join function.
55. *CO3 get drivers 2,* the Join function is used to extract receiver data from CO3 Receiver Data, that are specific to 2 level of cost object.
56. *CO3 allocate level 2,* the Allocation function is used to perform allocation costs to Cost Object 3, level 2, based on drivers generated in Join function.
57. *CO3 Get RC parent value,* the Join function is used to get parent values from *Parent-Child data* and CO3 Receiver data, in order to create all sibling values in next step.
58. *CO3 Get siblings,* the Join function is used to get parent values from *Parent-Child data* and *CO3 Get RC parent data*, in order to create all siblings for future generation of allocation drivers.
59. *CO3 get drivers level 1,* the Join function is used to generate drivers for allocation from CO3 Receiver Data, that are specific to level 1 of cost object.

60. *CO3 allocate level 1*, the Allocation function is used to perform allocation of costs to Cost Object 3, level 1, based on drivers generated in *CO3 get drivers level 1* Join function.
61. *CO3 execute allocation,* the Join function that is used to collect all allocation results from all levels per Cost Object 3 into one result.
62. *Join all results,* the Join function that is used to collect all allocation results of all three stages: Allocation to activities, Activity reallocation, Allocation to CO1, Allocation to CO2 and Allocation to CO3. This function is executable function that triggers all model.

Note

All these calculations are done and triggered as one process activity "Execute Calculation".

Please note as well that nowhere during the above calculation any aggregation happens or fields are taken out, in other words “no information is destroyed” and all results are available on granular level showing the complete traceability from the source of revenues and costs through all driver-based allocations down to the final result on product and service level.

3.3.1 Execute Hierarchy Format Change

This Join function is used to trigger writer *Initialize FHT* for initial data to be written in Flat hierarchies table and View function - *Hierarchies Iteration View* that contains iteration settings in order to generate all hierarchy levels.

As both the share the same fields and granularity in the rules of that function a simple union of both inputs is enough.

Note that no fields are taken out and all information is kept.

3.3.2 Flat Hierarchies Table

This Model Table function is used to store result of iterations and initial data set. Beside that it is used as input function in creating Iteration logic.

This function contains all fields that are related to generating all flat hierarchy levels.

3.3.3 Derive First 2 Hierarchy Levels

The Derivation function is used to derive first two levels of hierarchy using parent-child relationship from *Parent-Child Data* model table.

For demo purposes 7 rules are maintained and every rule contains existing Parent Value in Selection section and corresponding values that should be derived in Action section.

3.3.4 Initialize FHT

This Writer function is used to write result of *Derive first 2 hierarchy levels* function into *Flat* hierarchies table so the result is used later in processing, especially as input in creating iteration logic.

The Model writer type is set to 'Delete and Insert', so every time the function is run, data will be deleted and inserted again.

3.3.5 Hierarchies Iteration View

This View function is used to trigger iterations based on iterative logic that is created in other functions, in order to get all necessary hierarchy levels for further calculations.

As input function is used *Write to FHT* – writer function.

3.3.6 Get PC Data

This Join function is used to help in defining iteration logic and uses as inputs model table *Parent-Child Data* and *Flat hierarchies table.*

As Rule type ‘Inner Join’ is used, the join predicate is using Child field from Flat hierarchies table to be equal to Parent field from *Parent-Child Data model table.*

3.3.7 Derive Fields in FHT

The Derivation function is used to derive all children based on selected dimension in Selection section and it is used as function for generating iteration logic.

Derivation is performed through 98 rules in order to cover all possible combination of dimensions from selection section and children from action section.

3.3.8 Write to FHT

This Writer function is used as input for *Hierarchies Iterative View* to indirectly write results of iterations in *Flat hierarchies table* that are used in processing.

The Model writer type is set to ‘Insert’, so every time the function is run, data will be inserted again.

3.3.9 Responsibility Center Hierarchy Reformatted

This View function that is used after iteration process so specific dimensions could be extracted from *Execute hierarchy format change*, in this case *Responsibility centers*, and used in processing depending on different allocation levels.

View type is set to 'Explicit fields' so only fields from output will be shown in the results.

3.3.10 Activity Hierarchy Reformatted

This View function that is used after iteration process so specific dimensions could be extracted from *Execute hierarchy format change*, in this case *Activities*, and used in processing depending on different allocation levels.

View type is set to 'Explicit fields' so only fields from output will be shown in the results.

3.3.11 Cost Object 1 Hierarchy Reformatted

This View function that is used after iteration process so specific dimensions could be extracted from *Execute hierarchy format change*, in this case *Cost Object 1*, and used in processing depending on different allocation levels.

View type is set to 'Explicit fields' so only fields from output will be shown in the results.

3.3.12 Cost Object 2 Hierarchy Reformatted

This View function that is used after iteration process so specific dimensions could be extracted from *Execute hierarchy format change*, in this case *Cost Object 2*, and used in processing depending on different allocation levels.

View type is set to 'Explicit fields' so only fields from output will be shown in the results.

3.3.13 Cost Object 3 Hierarchy Reformatted

This View function that is used after iteration process so specific dimensions could be extracted from *Execute hierarchy format change*, in this case *Cost Object 3*, and used in processing depending on different allocation levels.

View type is set to 'Explicit fields' so only fields from output will be shown in the results.

3.3.14 Cost Object 4 Hierarchy Reformatted

This View function that is used after iteration process so specific dimensions could be extracted from *Execute hierarchy format change*, in this case *Cost Object 4*, and used in processing depending on different allocation levels.

View type is set to 'Explicit fields' so only fields from output will be shown in the results.

3.3.15 Cost Object 5 Hierarchy Reformatted

This View function that is used after iteration process so specific dimensions could be extracted from *Execute hierarchy format change*, in this case *Cost Object 5*, and used in processing depending on different allocation levels.

View type is set to 'Explicit fields' so only fields from output will be shown in the results.

3.3.16 Apply LIDACT Rules

This Join function that is used to combine *Line Item Values* and *Line Item Activity Rules* in order to get suitable sender data for next step in allocation to activities process.

This Join function is performed through 9 rules from left to right and defined complex predicates in order to ensure that applicable line is always taken.

3.3.17 Apply LIRD Rules

This Join function that uses *Apply LIDACT Rules* and *Line Item Resource Drivers* as inputs to consider all possible combinations in order to generate resource drivers since the combination of *Version*, *Period*, *Responsibility Center* and *Line Item* determine the *Resource Driver*.

This Join function is performed through 9 rules from left to right and defined complex predicates in order to ensure that applicable line is always taken.

3.3.18 Perform Allocation to Activities

This Allocation function is used to perform allocation of line item values to activities, since PCM concept considers allocation of costs to activities as first part of allocation process.

Therefore, we just maintain one rule here, by using a direct allocation (keeping all fields and thus providing maximum granular results). Receiver rule *Variable portion* and *Drivers Amount* as a base are used to allocate sender data.

3.3.19 Collect Activity Results

This Join function is used to collect all results from allocation and non-allocated costs. In following allocation segment, this function is used to generate sender data.

As both the share the same fields and granularity in the rules of that function a simple union of both inputs is enough.

3.3.20 Prepare Receiver at Lowest Granularity

This Join function is used to prepare receiver at lowest granularity since the drivers are usually attached at lowest granularity.

As input function are used *Activity Reallocation Data*, *Activity hierarchy reformatted* and *Responsibility Center* hierarchy reformatted and by inner join with all possible options defined in complex predicates it is ensured to get lowest granularity.

3.3.21 Get Drivers

This Join function is used to generate drivers that are used in reallocation of activities.

As input function are used *Resource Driver Split* and *Prepare receiver* at lowest granularity and by inner join with defined join predicates data is prepared for next stage of allocation.

3.3.22 Activity Reallocation

This Allocation function is used to perform reassignment of activities between each other. There are also activities that are non-reassign, and those will be allocated in future steps.

Activity reallocation is performed by iterative allocation with following settings:

- Cycle maximum Value: 15
- Result aggregation: None
- Activity Offset Sender Activity
- Responsibility Center Offset Sender Responsibility Center

Therefore, we just maintain one rule here, by using a direct allocation (keeping all fields and thus providing maximum granular results). Receiver rule *Variable portion* and *Driver Amount 1* as a base are used to allocate sender data.

3.3.23 Collect Activity Results

This Join function that uses as inputs *Activity Reallocation* and *Collect Activity Results* from previous allocation in order to prepare data for further allocation steps.

As both share the same fields and granularity in the rules of that function a simple union of both inputs is enough.

3.3.24 Sender Preparation - Apply COD Rules

This Join function that is used to determine sender data for allocation of costs on cost objects in combination with cost object drivers.

As input function are used *Collect Activity Results* and *Cost Object Drivers* and Join function is performed through 9 rules from left to right and defined complex predicates in order to ensure that applicable line is always taken.

3.3.25 Receiver Preparation - Apply COA Rules

This Join function is used to determine receiver data for allocation of costs on cost objects in combination with cost object allocation data.

As input function are used *Sender Prep - Apply COD Rules* and *Cost Object Allocation Data* and Join function is performed through 10 rules from left to right and defined complex predicates in order to ensure that applicable line is always taken.

3.3.26 CO1 Receiver Data

This Join function is used to determine receiver data for *Cost Object 1* using reformatted hierarchies in order to get receiver depending on level of cost object.

For demo purposes 3 rules are maintained. This function is used to inner join *Cost Object 1 hierarchy reformatted* with *Receiver Preparation – Apply COA rules*.

3.3.27 CO1 Get Drivers 3007

This Join function is used to extract receiver data from *CO1 Receiver Data*, that are specific to 3007 level of cost object.

For demo purposes 4 rules are maintained. This function is used to inner join *Activity Driver Values* with *CO1 Receiver Data*.

3.3.28 CO1 Allocate Level 3007

This Allocation function uses the *Sender prep - Apply COD rules* function as sender data and allocates it to *CO1 get drivers 3007* as receiver data. It performs allocation of costs to Cost object 1, level 3007, based on drivers generated in Join function.

Therefore, we just maintain one rule here, by using a direct allocation (keeping all fields and thus providing maximum granular results). Receiver rule *Variable portion* and *Driver Amount* as a base are used to allocate sender data.

3.3.29 CO1 Get Drivers 0

This Join function is used to extract receiver data from *CO1 Receiver Data*, that are specific to 0 level of cost object.

For demo purposes 4 rules are maintained. This function is used to inner join *Activity Drivers Values* with *CO1 Receiver Data*.

3.3.30 CO1 Allocate Level 0

This Allocation function uses the *Sender prep - Apply COD rules* function as sender data and allocates it to *CO1 get drivers 0* as receiver data. It performs allocation of costs to cost object 1, level 0, based on drivers generated in Join function.

Because it is using the result of another allocation as input, it can also be called a step-down or step-ladder allocation.

Therefore, we just maintain one rule here, by using a direct allocation (keeping all fields and thus providing maximum granular results). Receiver rule *Variable portions* and *Drivers Amount* as a base are used to allocate sender data.

3.3.31 CO1 Get Drivers 2

This Join function is used to extract receiver data from *CO1 Receiver Data*, that are specific to level 2 of cost object.

For demo purposes 4 rules are maintained. This function is used to inner join *Activity Driver Values* with *CO1 Receiver Data*.

3.3.32 CO1 Allocate Level 2

This Allocation function uses the *Sender prep - Apply COD rules* function as sender data and allocates it to *CO1 get drivers 2* as receiver data. It performs allocation of costs to cost object 1, level 2, based on drivers generated in Join function.

Because it is using the result of another allocation as input, it can also be called a step-down or step-ladder allocation.

Therefore, we just maintain one rule here, by using a direct allocation (keeping all fields and thus providing maximum granular results). Receiver rule *Variable portions* and *Driver Amount* as a base are used to allocate sender data.

3.3.33 CO1 Get RC Parent Value

This Join function is used to get parent values from *Parent-Child Data* and *CO1 Receiver Data*, in order to create all siblings' values in next step.

For demo purposes 3 rules are maintained. This function is used to left outer join *Parent-Child Data* with *CO1 Receiver Data* based on field *Child* as join predicate.

3.3.34 CO1 Get Siblings

This Join function is used to get parent values from *Parent-Child Data* and *CO1 Get RC parent Data*, in order to create all siblings for future generation of allocation drivers.

For demo purposes 3 rules are maintained. This function is used to inner join *Parent-Child Data* with *CO1 Get RC parent value*.

3.3.35 CO1 Get Drivers Level 1

This Join function is used to generate drivers for allocation from *CO1 Receiver Data*, that are specific to level 1 of cost object.

For demo purposes 3 rules are maintained. This function is used to inner join *CO1 Get siblings* with *Activity Driver Values.*

3.3.36 CO1 Allocate Level 1

This Allocation function uses the *Sender prep - Apply COD rules* function as sender data and allocates it to *CO1 get drivers level 1* as receiver data. It performs allocation of costs to cost object 1, level 1, based on drivers generated in Join function.

Because it is using the result of another allocation as input, it can also be called a step-down or stepladder allocation.

Therefore, we just maintain one rule here, by using a direct allocation (keeping all fields and thus providing maximum granular results). Receiver rule *Variable Portions* and *Driver Amount* as a base are used to allocate sender data.

3.3.37 CO1 Execute Allocation

This Join function is used to collect all allocation results from all levels into one result. Those values are used in future processing.

For demo purposes 5 rules are maintained. This function is used to union all *CO1 allocate level 3007* with *CO1 allocate level 0, CO1 allocate level 2* and *CO1 allocate level 1.*

3.3.38 CO2 Get Receiver Data

This Join function is used to determine receiver data for *Responsibility Center hierarchy reformatted* using reformatted hierarchies in order to get receiver depending on level of cost object.

For demo purposes 3 rules are maintained. This function is used to inner join *Cost Object 2 hierarchy reformatted* with *Receiver Preparation – Apply COA rules*.

3.3.39 CO2 Get Drivers 3007

This Join function is used to extract receiver data from *CO2 get receiver data*, that are specific to 3007 level of cost object.

For demo purposes 4 rules are maintained. This function is used to inner join *Activity Driver Values* with *CO2 get receiver data*.

3.3.40 CO2 Allocate Level 3007

This Allocation function uses the *CO1 execute Allocation function* as sender data and allocates it to *CO2 get drivers 3007* as receiver data. It performs allocation of costs to Cost object 2, level 3007, based on drivers generated in Join function.

Because it is using the result of another allocation as input, it can also be called a step-down or step-ladder allocation.

Therefore, we just maintain one rule here, by using a direct allocation (keeping all fields and thus providing maximum granular results). Receiver rule *Variable Portions* and *Driver Amount* as a base are used to allocate sender data.

3.3.41 CO2 Get Drivers 0

This Join function is used to extract receiver data from *CO2 get receiver data*, that are specific to 0 level of cost object.

For demo purposes 4 rules are maintained. This function is used to inner join *Activity Drivers Values* with *CO2 get receiver data*.

3.3.42 CO2 Allocate Level 0

This Allocation function uses the *CO1 execute Allocation function* as sender data and allocates it to *CO2 get drivers 0* as receiver data. It performs allocation of costs to cost object 2, level 3007, based on drivers generated in Join function.

Therefore, we just maintain one rule here, by using a direct allocation (keeping all fields and thus providing maximum granular results). Receiver rule *Variable Portions* and *Driver Amount* as a base are used to allocate sender data.

3.3.43 CO2 Get Drivers 2

This Join function is used to extract receiver data from *CO2 get receiver data*, that are specific to level 2 of cost object.

For demo purposes 4 rules are maintained. This function is used to inner join *Activity Drivers Values* with *CO2 get receiver data*.

3.3.44 CO2 Allocate Level 2

This Allocation function uses the *CO1 execute Allocation function* as sender data and allocates it to *CO2 get drivers 2* as receiver data. It performs allocation of costs to cost object 2, level 2, based on drivers generated in Join function.

Therefore, we just maintain one rule here, by using a direct allocation (keeping all fields and thus providing maximum granular results). Receiver rule *Variable Portions* and *Driver Amount* as a base are used to allocate sender data.

3.3.45 CO2 Get RC Parent Value

This Join function is used to get parent values from *Parent-Child data* and *CO2 get receiver data*, in order to create all siblings' values in next step.

For demo purposes 3 rules are maintained. This function is used to left outer join *Parent-Child Data* with *CO2 get receiver data* based on field *Child* as join predicate.

3.3.46 CO2 Get Siblings

This Join function is used to get parent values from *Parent-Child data* and *CO2 Ger RC parent data*, in order to create all siblings for future generation of allocation drivers.

For demo purposes 3 rules are maintained. This function is used to inner join *Parent-Child Data* with *CO2 Get RC parent value*.

3.3.47 CO2 Get Drivers Level 1

This Join function is used to generate drivers for allocation from *CO2 Receiver Data*, that are specific to level 1 of cost object.

For demo purposes 3 rules are maintained. This function is used to inner join *CO2 Get siblings* with *Activity Driver Values.*

3.3.48 CO2 Allocate Level 1

This Allocation function uses the *CO1 execute Allocation function* as sender data and allocates it to *CO2 get drivers level 1* as receiver data. It performs allocation of costs to cost object 2, level 1, based on drivers generated in Join function.

Therefore, we just maintain one rule here, by using a direct allocation (keeping all fields and thus providing maximum granular results). Receiver rule *Variable Portions* and *Driver Amount* as a base are used to allocate sender data.

3.3.49 CO2 Execute Allocation

This Join function is used to collect all allocation results from all levels into one result.

For demo purposes 5 rules are maintained. This function is used to union all *CO2 allocate level 3007* with *CO2 allocate level 0, CO2 allocate level 2 and CO2 allocate level 1.*

3.3.50 CO3 Get Receiver Data

This Join function is used to determine receiver data for *Responsibility Center hierarchy reformatted* using reformatted hierarchies in order to get receiver depending on level of cost object.

For demo purposes 3 rules are maintained. This function is used to inner join *Cost Object 3 hierarchy reformatted* with *Receiver Preparation – Apply COA rules*.

3.3.51 CO3 Get Drivers 3007

This Join function is used to extract receiver data from *CO3 get receiver data*, that are specific to 3007 level of cost object.

For demo purposes 4 rules are maintained. This function is used to inner join *Activity Driver Values* with *CO3 get receiver data*.

3.3.52 CO3 Allocate Level 3007

This Allocation function uses the *CO1 execute Allocation function* as sender data and allocates it to *CO3 get drivers 3007* as receiver data. It performs allocation of costs to Cost Object 3, level 3007, based on drivers generated in Join function.

Because it is using the result of another allocation as input, it can also be called a step-down or stepladder allocation.

Therefore, we just maintain one rule here, by using a direct allocation (keeping all fields and thus providing maximum granular results). Receiver rule *Variable Portions* and *Driver Amount* as a base are used to allocate sender data.

3.3.53 CO3 Get Drivers 0

This Join function is used to extract receiver data from *CO3 get receiver data*, that are specific to 0 level of cost object.

For demo purposes 4 rules are maintained. This function is used to inner join *Activity Drivers Values* with *CO3 get receiver data*.

3.3.54 CO3 Allocate Level 0

This Allocation function uses the *CO1 execute Allocation function* as sender data and allocates it to *CO3 get drivers 0* as receiver data. It performs allocation of costs to Cost Object 3, level 3007, based on drivers generated in Join function.

Therefore, we just maintain one rule here, by using a direct allocation (keeping all fields and thus providing maximum granular results). Receiver rule *Variable Portions* and *Driver Amount* as a base are used to allocate sender data.

3.3.55 CO3 Get Drivers 2

This Join function is used to extract receiver data from *CO3 get receiver data*, that are specific to level 2 of cost object.

For demo purposes 4 rules are maintained. This function is used to inner join *Activity Drivers Values* with *CO3 get receiver data*.

3.3.56 CO3 Allocate Level 2

This Allocation function uses the *CO1 execute Allocation function* as sender data and allocates it to *CO3 get drivers 2* as receiver data. It performs allocation of costs to Cost Object 3, level 2, based on drivers generated in Join function.

Therefore, we just maintain one rule here, by using a direct allocation (keeping all fields and thus providing maximum granular results). Receiver rule *Variable Portions* and *Driver Amount* as a base are used to allocate sender data.

3.3.57 CO3 Get RC Parent Value

This Join function is used to get parent values from *Parent-Child data* and *CO3 get receiver data*, in order to create all siblings' values in next step.

For demo purposes 3 rules are maintained. This function is used to left outer join *Parent-Child Data* with *CO3 get receiver data* based on field *Child* as join predicate.

3.3.58 CO3 Get Siblings

This Join function is used to get parent values from *Parent-Child data* and *CO3 Ger RC parent data*, in order to create all siblings for future generation of allocation drivers.

For demo purposes 3 rules are maintained. This function is used to inner join *Parent-Child Data* with *CO3 Get RC parent value*.

3.3.59 CO3 Get Drivers Level 1

This Join function is used to generate drivers for allocation from *CO3 Receiver Data*, that are specific to level 1 of cost object.

For demo purposes 3 rules are maintained. This function is used to inner join *CO3 Get siblings* with *Activity Driver Values.*

3.3.60 CO3 Allocate Level 1

This Allocation function uses the *CO1 execute Allocation function* as sender data and allocates it to *CO3 get drivers level 1* as receiver data. It performs allocation of costs to Cost Object 3, level 1, based on drivers generated in Join function.

Therefore, we just maintain one rule here, by using a direct allocation (keeping all fields and thus providing maximum granular results). Receiver rule *Variable Portions* and *Driver Amount* as a base are used to allocate sender data.

3.3.61 CO3 Execute Allocation

This Join function is used to collect all allocation results from all levels into one result.

For demo purposes 5 rules are maintained. This function is used to union all *CO3 allocate level 3007* with *CO3 allocate level 0, CO3 allocate level 2 and CO3 allocate level 1.*

3.3.62 Join All Results

This Join function is used to collect all allocation results from all stages of allocation. This function is executable function, used to trigger all model.

For demo purposes 5 rules are maintained. This function is used to union all *Collect activity results* with *Activity reallocation, CO1 execute allocation, CO2 execute allocation and CO3 execute allocation.*

3.4 Reporting

In this section, additional Query functions for reviewing results by execution users are defined.

In the process these review and reporting activities happen, after the Profitability and Cost Management Calculation is executed.

- *Activity Results*: Focuses on the results at activity level.
- *Activity Results [Side-by-side Simulation]*: Focuses on the results at activity level and used for simulation purpose in Qualitative report
- *Activity Reallocation Results:* This Query function provides results of activity reassignment, performed by iterative allocation
- *Cost Object 1 Results*: Provides results at cost object 1 level
- *Cost Object 2 Results*: Focuses on the results at cost object 2 level
- *Cost Object 3 Results*: Focuses on the results at cost object 3 level
- *Activity and Cost Object Results:* This Query function provides review results of cost objects and contribution of each activity

> Note
>
> Specific chart types have not been defined in this sample content. Reports, therefore, use the "Column" chart type. However, end users can change this type on the fly and save their chart type as the default layout.

3.4.1 Activity Results

This Query function is used to provide results at per each activity level.

The following fields are preconfigured in the report:

- *Amount*
- *Activity*
- *CO1 Driver Level*
- *Line Item*
- *Cost Object 1*
- *Cost Object 2*

3.4.2 Activity Results [Side-by-side Simulation]

This Query function is used to provide results at per each activity level. This query is used for purpose of simulation.

The following fields are preconfigured in the report:

- *Amount*

- *Activity*
- *CO1 Driver Level*
- *Line Item*
- *Cost Object 1*
- *Cost Object 2*

3.4.3 Activity Reallocation Results

This Query function provides results of activity reassignment, performed by iterative allocation.

The following fields are preconfigured in the report:

- *Amount*
- *Activity*
- *Responsibility Center*
- *Sender Responsibility Center*
- *Line Item*
- *Function ID*
- *Period*

3.4.4 Cost Object 1 Results

This Query function provides read data access to Responsibility Center Results per each activity.

The following fields are preconfigured in the report:

- *Amount*
- *Activity*
- *Responsibility Center*
- *Line Item*
- *Cost Object 1*
- *Cost Object 2*
- *Period*
- *Function ID*

3.4.5 Cost Object 2 Results

This Query function provides read data access to Responsibility Center Results per each activity.

The following fields are preconfigured in the report:

- *Amount*
- *Activity*
- *Responsibility Center*
- *Line Item*
- *Version*
- *Cost Object 2*

- *Period*
- *Function ID*

3.4.6 Cost Object 3 Results

This Query function provides read data access to Responsibility Center Results per each activity.

The following fields are preconfigured in the report:

- *Amount*
- *Activity*
- *Responsibility Center*
- *Line Item*
- *Version*
- *Cost Object 3*
- *Period*
- *Function ID*

3.4.7 Activity and Cost Object Results

This Query function provides read data access to results of cost objects and contribution of each activity.

The following fields are preconfigured in the report:

- *Amount*
- *Activity*
- *Function ID*
- *Responsibility Center*
- *Line Item*
- *Period*
- *Version*
- *Cost Object 1*
- *Cost Object 2*
- *Cost Object 3*

Important Disclaimers and Legal Information

Hyperlinks

Some links are classified by an icon and/or a mouse over text. These links provide additional information.
About the icons:

- Links with the icon  : You are entering a Web site that is not hosted by SAP. By using such links, you agree (unless expressly stated otherwise in your agreements with SAP) to this:
    - The content of the linked-to-site is not SAP documentation. You may not infer any product claims against SAP based on this information.
    - SAP does not agree or disagree with the content on the linked-to-site, nor does SAP warrant the availability and correctness. SAP shall not be liable for any damages caused by the use of such content unless damages have been caused by SAP's gross negligence or willful misconduct.
- Links with the icon  : You are leaving the documentation for that particular SAP product or service and are entering a SAP-hosted Web site. By using such links, you agree that (unless expressly stated otherwise in your agreements with SAP) you may not infer any product claims against SAP based on this information.

Beta and Other Experimental Features

Experimental features are not part of the officially delivered scope that SAP guarantees for future releases. This means that experimental features may be changed by SAP at any time for any reason without notice. Experimental features are not for productive use. You may not demonstrate, test, examine, evaluate or otherwise use the experimental features in a live operating environment or with data that has not been sufficiently backed up.

The purpose of experimental features is to get feedback early on, allowing customers and partners to influence the future product accordingly. By providing your feedback (e.g. in the SAP Community), you accept that intellectual property rights of the contributions or derivative works shall remain the exclusive property of SAP.

Example Code

Any software coding and/or code snippets are examples. They are not for productive use. The example code is only intended to better explain and visualize the syntax and phasing rules. SAP does not warrant the correctness and completeness of the example code. SAP shall not be liable for errors or damages caused by the use of example code unless damages have been caused by SAP's gross negligence or willful misconduct.

Gender-Related Language

We try not to use gender-specific word forms and formulations. As appropriate for context and readability, SAP may use masculine word forms to refer to all genders.

www.sap.com