User Guide | PUBLIC
Document Version: 3.0 SP22 – 2023-06-20

SAP Profitability and Performance Management

THE BEST RUN SAP

Content

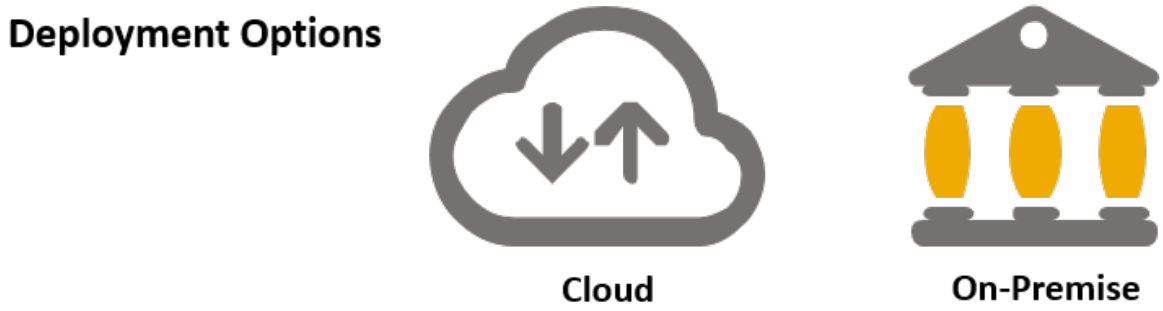
1 SAP Profitability and Performance Management

With SAP Profitability and Performance Management, SAP provides a new generation of integrated performance management applications that do not require their own data model but can use and reuse existing data and information models from other SAP and non-SAP applications in the cloud or on-premise.

SAP Profitability and Performance Management is built on the in-memory platform SAP HANA. Using the advanced potential of SAP HANA, SAP Profitability and Performance Management is designed for business and provides an instant insight by using a single source of truth, real-time processes, and agile financial and business modeling capabilities. Thanks to the principles of SAP Fiori user experience, it is designed to run simply and comfortably for business users.

Implementation Considerations

SAP Profitability and Performance Management can be deployed both in the cloud and on-premise and covers various integration scenarios.



SAP Profitability and Performance Management can be used on a separate instance and can still be integrated with other SAP and non-SAP components. We recommend that you implement SAP Profitability and Performance Management as closely as possible to the source data. If other applications that contain relevant data are already installed on SAP HANA, we recommend that you use SAP Profitability and Performance Management on the same SAP HANA platform or even on the same instance to ensure optimal performance and the maximum reuse of existing data and metadata, such as hierarchies, master data, and so on.

Integration

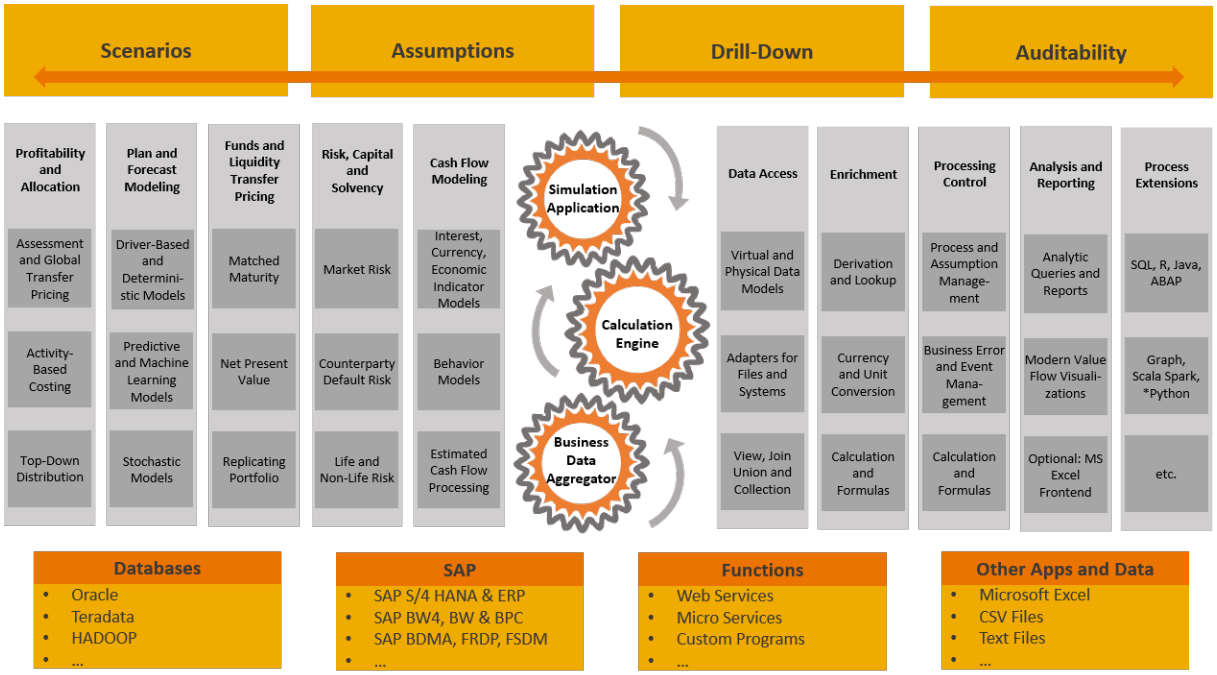
The business data aggregation capabilities of SAP Profitability and Performance Management (abbreviated as “PaPM” in the diagram below) enable the integration of operational systems and data warehouses at high speed with little or no data replication.

SAP Profitability and Performance Management uses the official application interfaces from the SAP or non-SAP applications for data read access, for example CDS views, from SAP HANA or SAP S/4HANA, open ODS views from SAP BW or BW/4HANA, calculation views from CAR, FSDM and BDMA – either locally or remotely via smart data access. If this redundancy-free approach is not feasible, SAP Profitability and Performance Management uses other official application interfaces, such as SAP BAPIs and Web services, or classic file imports of various formats.

SAP Profitability and Performance Management uses the official application interfaces from the SAP or non-SAP application for data write access, for example SAP HANA-based write interfaces like HAP to BW or BW/4HANA, SAP HANA-based PAK functions to BPC and AMDP interfaces to the Results Data component. If this redundancy-free approach is not feasible, SAP Profitability and Performance Management uses other official application interfaces, such as SAP BAPIs and Web services, or classic file exports of various formats.

Features

The simulation application capabilities of SAP Profitability and Performance Management enable the execution of what-if scenarios for business users and the management of assumptions and drivers. Based on the granularity of the financial model, it allows drill-down from high-level to very detailed results and provides transparency by offering traceability and auditability information. In addition, it allows non-SAP and SAP BI tools, like SAP Analysis for Microsoft Office, to access the information or even trigger further calculations.



The calculation engine of SAP Profitability and Performance Management allows business users to design and execute financial and business models by configuring and combining functions across the following areas:

1. Profitability and allocation
    1. Global transfer pricing
    2. Assessments
    3. Activity-based costing
    4. Top-down distribution
2. Plan and forecast modeling
    1. Driver-based and deterministic models
    2. Predictive and machine learning models
    3. Stochastic models
3. Funds and liquidity transfer pricing
    1. Matched maturity approach
    2. Net present value approach
    3. Replication portfolio approach
    4. Further volume and account-based methods
4. Risk, capital, and solvency
    1. Market risk calculations
    2. Counterparty default risk calculations
    3. Life and non-life risk calculations
5. Cash flow modeling
    1. Interest, currency, and economic indicator models
    2. Behavior models
    3. Estimated cash flow processing
6. Data access
    1. Access to local and remote virtual and physical data models
    2. Adapters for files and selected systems
    3. Views, joins, unions and collections
7. Enrichment
    1. Derivations and lookups
    2. Currency and unit conversions
    3. Calculations and formulas
8. Processing control
    1. Process and assumption management
    2. Business error and event management
    3. Report management
9. Analysis and reporting
    1. Analytic queries and reports
    2. Item variance and reconciliation reports
    3. Optional SAP Analysis for Microsoft Office/Excel frontend
10. Process extensions
    1. SQL, R, Graph Script, Scala Spark
    2. Java and ABAP
    3. Further custom programs via industry standard interfaces like Web services

Related Information

See also Applications for Business Users [page 32]

For more information about the available functions in SAP Profitability and Performance Management, see Modeling Environment [page 48].

1.1 Main Use Cases and Sample Content

SAP Profitability and Performance Management is a native digital performance management solution that maintains and executes complex calculations, rules and simulations. Built on SAP HANA, it 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.

Sample content comprises configuration examples made available to customers to demonstrate best practices and ideas on how to model a use case using SAP Profitability and Performance Management.

SAP Profitability and Performance Management provides the following types of sample content:

- Cross-industry sample content
- Industry-specific sample content

Additional Information

For more information about the available sample content, see Sample Content.

1.2 Concepts for Key Users

Get an overview of the general concepts and integration capabilities on which SAP Profitability and Performance Management is built.

The following concepts are relevant for key users to help them understand how SAP Profitability and Performance Management works and how it can be used.

1. Financial and Business Modeling Entities
   SAP Profitability and Performance Management uses entities like environments, calculaton units and functions to structure and simplify the design of financial and business models, irrespective of the specific purpose and across business areas such as controlling, finance or risk.
2. Function Building Blocks and reusable Templates
   The functions use a common building block approach so that they can be plugged together to work in a common financial model. Each of these functional building blocks are systematically designed to be available and visible for use in every function only as necessary. In a general context, these function building blocks comprise header, input, lookup, signature, rules, checks and documentation.

3. Information Models for Business Entity Master Data and Lookup
   Data model functions can be used to make central master data information available to all functions via lookup formulas.
4. Parallelization and Partitioning
   By default, SAP Profitability and Performance Management takes care of runtime optimization automatically. For high-end computing requirements, you can make manual parallelization and partitioning settings to optimize the runtime further.
5. Roles and Authorizations
   SAP Profitability and Performance Management allows you to manage authorizations based on applications and functions. You can also set up characteristic-based authorizations for data to restrict the visibility of data.
6. Integration with non-SAP Systems and File Import/Export
   You can integrate SAP Profitability and Performance Management with non-SAP systems using various industry standards.
7. Integration with BW, BPC and AfO
   SAP Profitability and Performance Management allows integration with SAP Business Warehouse, SAP Business Planning and Consolidation and SAP Analysis for Office, including redundancy-free reuse of data, master data and hierarchies.
8. Integration with ERP and S/4HANA
   SAP Profitability and Performance Management allows integration with ERP and SAP S/4HANA, including redundancy-free reuse of data, master data and hierarchies as well as allocation rules.
9. Integration with SAP Analytics Cloud and SAP Digital Boardroom
   SAP Profitability and Performance Management allows integration with SAP Digital Boardroom and SAP Analytics Cloud using live data and imported data connections.
10. Integration with FRDP
    SAP Profitability and Performance Management allows integration with Finance and Risk Data Platform (FRDP), including redundancy-free reuse of data, master data and hierarchies, as well as the CVPM process orchestration of SAP Profitability and Performance Management functions and fast Results Data storage.

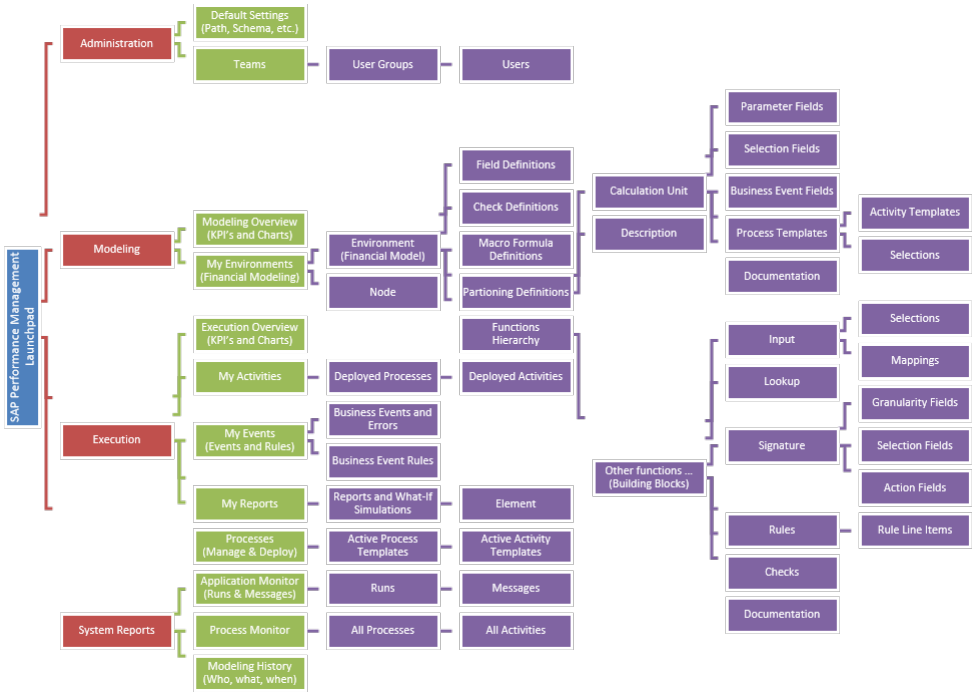
Related Information

For more information about SAP Profitability and Performance Management functions see Functions [page 107].

1.2.1 Financial and Business Modeling Entities

Get an overview of the most important entities in SAP Profitability and Performance Management and where to find them.

SAP Profitability and Performance Management uses entities to structure, harmonize and simplify the design of a financial and business model irrespective of its purpose (for example, controlling, finance or risk).



Financial and Business Modeling Entities

The picture gives an overview of these entities, how they relate to each other and where to find them on the user interface of the respective application.

Users can view or edit all or parts of the above entities depending on the authorizations and roles they have been assigned.

Related Information

For more information about SAP Profitability and Performance Management functions see Functions [page 107].

For more information about entities see Applications for Business Users [page 32].

1.2.2 Function Building Blocks

Function building blocks are the basis on which SAP Profitability and Performance Management functions are built. This allows functions to be connected to each other to design comprehensive financial and business models, and to fulfill complex activities in end-to-end processes. It is also the basis for incorporating reusable function templates which can reduce the configuration effort.

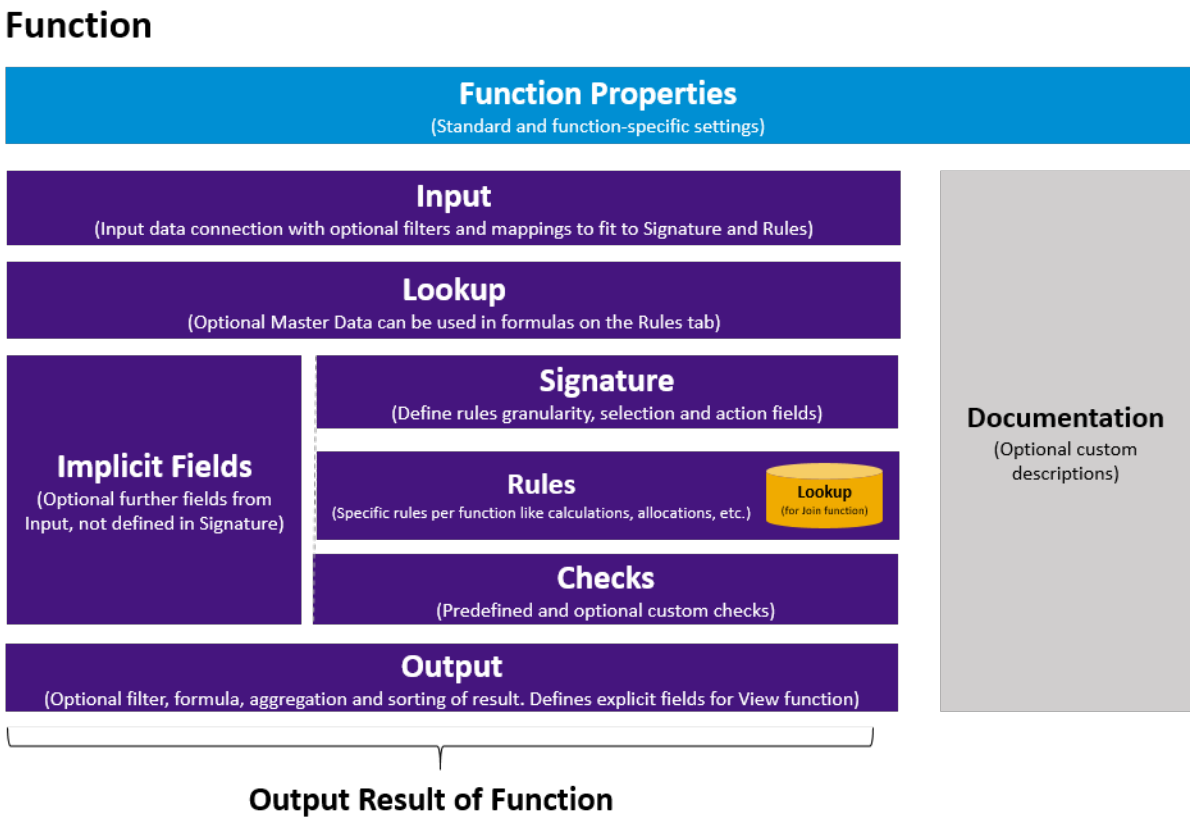
According to their technical content and specific use, functions can be classified in different categories. The following function categories are available:

- Information Functions [page 109]

- Processing Functions [page 138]
- Write and Adapter Functions [page 302]
- Structuring Functions [page 337]
- Query Function [page 347]
- Analytics Function [page 361]

Depending on the categories, some or all of the building blocks are relevant for a given function.

The diagram below provides a general overview of the function building blocks that appear as tabs on the function details user interface.



The following function building blocks are available:

- Header [page 10]
- Input [page 11]
- Lookup [page 12]
- Signature [page 13]
- Output [page 15]
- Rules [page 16]
- Checks [page 17]

Related Information

For more information about SAP Profitability and Performance Management functions see Functions [page 107].

1.2.2.1 Header

The header belongs to the individual part of a function. Wherever functionally feasible, it contains the following common standard settings:

1. *Include original Input Data*
   If you select “Yes”, the original input data is added to the output data along with the produced results. This makes it easier to model requirements, in cases where one scenario is built on top of another and the original scenario results therefore need to be kept and addition results need to be added to them.
2. *Suppress initial Results*
   If you select “Yes”, results that contain only their initial values in their action fields (for example, key figures 0 and characteristics “ ”) are excluded from the output. If initial result records have no effect on the result, it can reduce the data volume and processing of unnecessary records.
3. *Result Model Table*
   If no model table is assigned, the function is directly executed in the modeling environment and the results are stored in a function-specific temporary table. Otherwise, the results of the function are stored in the model table.
   The following scenarios apply:
   - Fields are in the results of a function but not included in the result model table, these fields are not automatically included or disregarded. During *Run* or *Show*, the function result adapts the fields and the data available in the result model table.
   - A function uses an input function with the result model table, the function consumes the data stored in the result model table and uses all the fields.
   - A function uses an input function with the result model table and the processing type is set to “sub-function”, the data is not written in the result model table unless the function is assigned as an activity under a process template or a run has been initiated.
   - A function uses an input function with the result model table and the processing type is set to “executable”, the data is written in the result model table when the function is executed.
4. *Result Handling*
   This setting offers various options:
   1. “Include enriched data”: This setting includes only data records in the results to which a rule was applied. At the end of the function, if there are data records for which the system was unable to apply a rule, it writes a warning to the message log.
   2. “Include all data”: This setting includes all data records in the results, irrespective of whether a rule was applied or not.
   3. “Error on non-enriched data”: This setting works in the same way as for include enriched data. However, for non-enriched data an error message is prepared and further processing is done based on the function event type setting:
      1. “Logging”: The error is written to the message log.
      2. “Management”: The error is written to the message log and a business event is registered so that the business user can deal with the exceptional situation and fix it.

4. “Abort on non-enriched data”: This setting works in the same way as errors in non-enriched data, but instead of an error message the system writes an abort message to the log, and the function is terminated.

Procedure

As each function has different header fields, you can find more information about the specific header procedures in the *Functions* section (see Functions [page 107]).

1.2.2.2 Input

All processing functions, write and adapter functions as well as query functions have an input. The input connects the function with a preceding function.

You can configure specific selections to restrict the data transferred from the *Input* function, and can configure mappings to adapt the data to the required signature of the function. The latter also helps if the function rules are based on a function template and different data with different fields needs to be processed based on common rules.

In contrast to the data that comes from lookup, the data that comes from the input is the basis for the business event and error management, including partial restart capabilities.

Procedure

Adding an Input Function

Once you have defined the function properties, you can add an input function to the function that you want to set up. To do so, follow the steps below:

1. Go to the *Input* tab.
2. Open the dropdown list of the *Input Function* field or choose *Function Select Open* to see all available input functions.
3. Select the input function you want to add.
4. Choose *Save*.

Applying a Formula to a Field (Optional)

Once you have defined the input function, you can set up or apply a formula for specific fields. To do so, follow the steps below:

1. Go to the *Input* tab and choose + (*Add*).
2. On the *Add Fields* screen, select all fields to which you want to apply formulas.
3. Choose *OK*.
4. On the next screen, select a specific field and choose *f* (*Formula*).

5. The system displays the *Formula* screen where you can now enter the formula.
6. If you have added multiple fields in step 2, repeat step 3 and 4 for all of these fields.

> **i Note**
>
> Alternatively, you can press CTRL and select all assigned fields at once in step 3. In this case, the system asks you to choose one of the selected fields on the *Formula* screen in step 4 before you can enter the formula.

7. Choose *OK*.
8. Choose *Save*.

Applying a Selection Condition to a Field (Optional)

Once you have defined the input function, you can set up or apply a selection condition for specific fields. To do so, follow the steps below:

1. Go to the *Input* tab and choose + (*Add*).
2. On the *Add Fields* screen select all fields for which you want to apply selection conditions.
3. Choose *OK*.
4. On the next screen, select a specific field and choose *Selection Condition*.
5. The system displays the *Selection Condition* screen where you can now enter the selection condition.
6. If you have added multiple fields in step 2, repeat step 3 and 4 for any of these fields.

> **i Note**
>
> Alternatively you can press CTRL and select all assigned fields at one time in step 3. In this case the system asks you to choose one of the selected fields on the *Selection Condition* screen in step 4 before you can enter the selection condition.

7. Choose *OK*.
8. Choose *Save*.

Removing Fields

To remove a field from the *Input* tab, follow the steps below:

1. Select the field you want to remove.
2. Choose − (*Remove Field*).
3. Choose *Save*.

1.2.2.3 Lookup

In the *Calculation*, *Funds Transfer Pricing* and *Valuation* functions, you can use lookup data models, which you can access in formulas to look up central master data settings.

To be able to use lookup data models in formulas, you first need to ensure they are registered on the *Lookup* tab.

Procedure

Adding Functions

Once you have defined the function properties, you can add one or more functions to the *Lookup* tab of the function that you want to set up. To do so, follow the steps below:

1. Go to the *Lookup* tab and choose + (*Add*).
2. Select the function to be added.
3. Choose *OK*.
4. Choose *Save*.

Removing a Function

To remove a function from the *Lookup* tab, follow the steps below:

1. Select the function you want to remove.
2. Choose — (*Remove*).
3. Choose *Save*.

1.2.2.4 Signature

All processing functions have a signature, which can produce a result for subsequent functions. The signature defines the minimum number of relevant fields of a function. There can also be further implicit fields from the input. These simply pass through the function without any change or any effect on the logic, and also appear in the output if no aggregation within the function is defined. If you add or remove fields from the data model this implicit field handling ensures the following:

- Data model changes do not affect the calculation model as long as no signature field is removed.
  If signature fields are missing, the input needs to be adapted to provide another field or mapping or a formula to substitute the original field. Alternatively, the rules of the function need to be adapted.
- Data model changes are propagated through all subsequent functions of the model automatically.
  The only exception is if a function uses explicit field handling to explicitly define the fields of the output, for example, in a view or if a rule includes aggregation (grouping).
- Data model changes are automatically propagated to queries for reporting.
  If a field is added, it is available for reporting. If a field is removed, it is no longer available for reporting. The latter can have an effect on predefined layouts, which then look different and might need to be adjusted.

The only functions that offer explicit field handling are views and joins. Aggregations can be run based on specific configuration settings and rules in the following functions:

- *Allocation*
- *Flow Modeling*
- *Funds Transfer Pricing*
- *Join*
- *Transfer Structure*
- *Valuation*

The signature is the interface of a function and defines a simple pivot table, in which calculations and logic can be applied. The signature is structured into three groups of fields, on which computations and modifications can occur:

- Header fields that describe the granularity characteristics.
- Row fields that describe the selection characteristics.
- Value fields that describe the action key figures and characteristics.

Granularity Fields	**VERSION**	Actual		
Selection Fields	**COST_CENTER**	**COST_ELEMENT**	**AMOUNT**	**QUANTITY**	*Action Fields*
	Cost Center 1	Cost Element 1	100	1	
	Cost Center 2	Cost Element 2	200	2	
	Cost Center 3	Cost Element 3	300	3	
	Cost Center 4	Cost Element 4	400	4	
	Cost Center 5	Cost Element 5	500	5	
	Cost Center 1	Cost Element 6	600	6	
	Cost Center 2	Cost Element 7	700	7	
	Cost Center 3	Cost Element 8	800	8	
	Cost Center 4	Cost Element 9	900	9	
	Cost Center 5	Cost Element 10	1000	10	

Signature

The figure contains an example where the granularity fields contain the functional area, the selection fields contain the cost center plus cost element and the action fields contain the amount and quantity.

More details regarding these three groups of fields in a signature are described below:

1. Granularity Fields
   Granularity fields define the minimum granularity of a function. They cannot occur as selection or action fields in the same function, which means rules or modifications on the granularity fields are not allowed. Instead, the granularity fields always stay stable from input through processing until output of the function. In data warehouses they are also known as block characteristics. Formulas and formula functions, like aggregations including SAP HANA window functions, are not allowed across values of these characteristics. Granularity fields can be used for horizontal package parallel processing, because they ensure that the overall result is always the same, irrespective of whether all the data is processed in one or multiple packages when you use granularity fields for grouping. A typical example is the granularity field “VERSION” in a calculation function, which ensures that all calculations are run for each version and not across all versions.
2. Selection Fields
   Selection fields can be used as a condition within the rules of a function. A typical example is the selection field “COST_ELEMENT” in a calculation, which allows the rules to be applied to selected cost elements only.
3. Action Fields
   Action fields can be used for calculation formulas and assignments within the rules of a function because their values can be changed in the function. A typical example is the action field “AMOUNT QUANTITY” in an allocation, which can then be allocated and distributed.

Selection and action fields can also overlap in certain functions. For example, a cost center can be used in a derivation both as a selection and an action field to fill in a default cost center value if the original value is empty.

> **i** Note
>
> You can drag and drop the fields between *Granularity*, *Selection* and *Action* field lists.

Procedure

Adding Fields

Once you have defined the function properties, you can add fields to the *Granularity*, *Selection* and *Action* field lists of the function that you want to set up. To do so, follow the steps below:

1. Go to the *Signature* tab and choose + (*Add*).
2. Select the fields to be added.
3. Choose *OK*.
4. Choose *Save*.

Removing Fields

To remove a field from the *Signature* tab, follow the steps below:

1. Select the field you want to remove.
2. Choose — (*Remove Field*).
3. Choose *Save*.

1.2.2.5 Output

In the *Writer* function and *View* functions, you can define additional field details (such as selection conditions, formulas, group aggregations and sort orders) on the *Output* tab.

This is mandatory for the *View* function with type “Explicit Fields”.

> **i** Note
>
> The fields defined on the *Signature* tab are also available on the *Output* tab.

Procedure

Adding Fields

Once you have defined the function properties, you can add fields to the *Output* tab of the function that you want to set up. To do so, follow the steps below:

1. Go to the *Output* tab and choose + (*Add*).
2. Select the fields to be added.
3. Choose *OK*.
4. Choose *Save*.

Applying a Formula to a Field (Optional)

To set up or apply a formula to a specific field, follow the steps below:

1. Select the field, as required, and choose *f* (*Formula*).
2. The system displays the *Formula* screen where you can now enter the formula.
3. Choose *OK*.
4. Choose *Save*.

Applying a Selection Condition to a Field (Optional)

To set up or apply a selection condition to a specific field, follow the steps below:

1. Select the field, as required, and choose *Selection Condition*.
2. The system displays the *Select Condition* screen where you can now enter the condition.
3. Choose *OK*.
4. Choose *Save*.

Removing Fields

To remove a field from the *Output* tab, follow the steps below:

1. Select the field you want to remove.
2. Choose − (*Remove Field*).
3. Choose *Save*.

1.2.2.6 Rules

Rules contain the individual part of most of the functions. They contain the following common fields:

1. *Rule ID*
   The rule ID has to be unique in a function. If (interim) results of a function are persisted, the rule ID is stored to enable you to trace which rule of a function was applied to each data record.
2. *Rule State*
   The rule state can be active or inactive. Inactive rules are not executed. For example, you can set a rule to inactive if you temporarily do not want the system to apply it. You do not need to delete the rule and reenter it again later.
3. *Rule Level*

You can use the rule level to define hierarchical rules.

4. *Rule Description*
   You can use the rule description to enter a user-defined text and comments.

Procedure

As each function has its own ruling behavior, you can find more information about the specific rules in the *Functions* section (see Functions [page 107]).

1.2.2.7 Checks

You can run custom checks on the results data of all processing functions, and of write and adapter functions.

Checks are defined at environment level and can be registered in one or more functions. When a function is executed, these checks are applied to the result of the function. If the check condition is satisfied, an appropriate message is written to the application log.

If the business event and error management is activated for the function and the message type is either “error” or “abort”, business events are also created. You can deal with these business events in the *My Events* application.

> **i Note**
>
> Unlike other business functions, checks are working differently with Workbook Adapter and Remote Function Adapter. Checks are applied to input data instead of result data. You can run custom checks on the input data of Workbook Adapter and Remote Function Adapters.

Procedure

To add checks to the *Checks* tab of the function being setup, follow the steps below after you have defined the function properties:

1. Go to the *Checks* tab and choose + (*Add*).
2. Select the check to be used.

   > **i Note**
   >
   > Please add the selected check also to the environment details. For more information about environment details, see Environment Details [page 49].

3. Choose *OK*.
4. Choose *Save*.

1.2.3 Information Models for Master Data and Lookup

The term “Master Data” is used in the following two ways:

1. **Master Data of a Field**
   Field master data defines the values permitted for a field like InfoObjects and data elements. For InfoObjects, you can also define hierarchies on top to structure the permitted values further for calculation and reporting.
2. **Master Data of a Business Entity**
   Business entity master data defines a table or a set of a tables used to define records with combinations of characteristic and key figure values according to the business requirements, like product master data, financial instrument master data and so on. Business entity master data is rarely changed and is reused by many functions to control calculation (for example, how the funds transfer price of a retail loan is calculated).
   Business entity master data can reside in any model function.

Both kinds of master data are supported by SAP Profitability and Performance Management and can be used for lookup.

Usage and Lookup

The usage and lookup of master data happens in two steps.

1. The respective model functions needs to be registered on the *Lookup* tab. These model functions then contain the master data. To use the master data for further processing, a lookup ID has to be defined.
2. The lookup of data can then be included in a formula. The format for lookup consists of the lookup ID followed by the field to be looked up and then square brackets, in which the selections are defined.

If multiple records fulfill the lookup criteria, the default aggregation is used to return exactly one value.

Example

The following master data is available under lookup ID `MY_DATA`.

Example of Master Data

COST_CENTER	COST_ELEMENT	AMOUNT	QUANTITY
Cost Center 1	Cost Element 1	100	1
Cost Center 2	Cost Element 2	200	2
Cost Center 3	Cost Element 3	300	3
Cost Center 4	Cost Element 4	400	4
Cost Center 5	Cost Element 5	500	5
Cost Center 1	Cost Element 6	600	6

COST_CENTER	COST_ELEMENT	AMOUNT	QUANTITY
Cost Center 2	Cost Element 7	700	7
Cost Center 3	Cost Element 8	800	8
Cost Center 4	Cost Element 9	900	9
Cost Center 5	Cost Element 10	1000	10

This results in the following output:

- The lookup statement **`MY_DATA.AMOUNT[COST_ELEMENT='Cost Element 1']`** would return the amount "100".
- The lookup statement **`MY_DATA.QUANTITY[COST_ELEMENT='Cost Element 1']`** would return the quantity "1".
- The lookup statement **`MY_DATA.COST_ELEMENT[AMOUNT=500]`** would return the cost element "Cost Element 5".
- If the default aggregation for the field *Amount* is summation, the lookup statement **`MY_DATA.AMOUNT[COST_CENTER='Cost Center 1']`** would return the amount "700", because the value "Cost Center 1" is not unique and the amount is therefore added up to `100+600 => 700` automatically.
- If the default aggregation for the field *Cost Element* is maximum, the lookup statement **`MY_DATA.COST_ELEMENT[COST_CENTER='Cost Center 2']`** would return the cost element "Cost Element 7", because the value "Cost Center 2" is not unique and the cost element maximum (which is "Cost Element 7") is therefore taken automatically.
- The lookup statement **`MY_DATA.AMOUNT[COST_ELEMENT='ABC']`** would return the amount "0", which is the initial value of the field *Amount*, because there is no cost element "ABC" in the master data.
- The lookup statement **`MY_DATA.COST_CENTER[COST_ELEMENT='ABC']`** would return the cost center " ", which is the initial value of the field *Cost Element*, because there is no cost element "ABC" in the master data.

Related Information

For more information about SAP Profitability and Performance Management functions, see Functions [page 107].

1.2.4 Activation of Functions, Process Templates and Environments

SAP Profitability and Performance Management clearly separates the design of a model from its execution. A model is designed in the modeling environment application. This is sometimes also referred to as Customizing. The *Activate* button in the modeling environment is used by the modeling user to trigger the generation of all the required artifacts once a function or a process is designed and ready. This activation is a mandatory step that ensures that the function or process template is ready to be executed.

The following *Activate* buttons exist:

1. *Activate* button in the *Calculation Unit* function
   This activation goes through the entire environment and activates everything that is required. This means that afterwards in the processes application, new processes with activities can be deployed and execution users can work on these processes and activities. If you set up a new process template or change an existing process template configuration and its underlying activities, this activation has to be triggered.
2. *Activate* button for individual function
   This activation activates an individual function, including any required sub-functions and underlying data model functions. The main purpose here is to allow the modeling user to test and run the function directly from within the modeling environment by choosing the *Run* button for that particular function.
3. *Activate* button in function hierarchy
   If the modeling user has selected multiple functions in the function hierarchy, this *Activate* button calls the activation for every function that has been selected. The main purpose is the same as for the individual function.

Related Information

For more information about SAP Profitability and Performance Management, see Functions [page 107].

1.2.5 Parallelization and Partitioning

Partitioning defines and enables the parallel package processing of data. Partitioning splits the dataset into subsets and triggers the calculation logic for several parallel threads at the same time.

For high-end scenarios, you need to explicitly configure parallelization and partitioning in the modeling environment to enable you to do the following:

1. Handle datasets with more than 2 billion records
   If the data volume of a function exceeds 2 billion records, partitioning and parallelization must be set up so that the volume of each partition is below 2 billion records.
2. Actively manage RAM and CPU usage
   If the usage of RAM and CPU resources during execution needs to be restricted, you can set up partitioning and parallelization so that only a subset of data is processed at the same time.

In both scenarios, the dataset has to be logically separated into parts that can be processed independently of other parts.

1.2.5.1 Partitioning and Range Setup

Partitioning Setup

Follow the steps below to set up the partitioning:

1. On the *Modeling Environment* screen, choose the *Environment* button located in the screen header buttons section. The system opens the *Environment Details* section on the right-hand side of the screen.
2. Go to the *Partitioning* tab and choose the *Edit* button located in the upper right corner of the screen, then choose *Add*.
3. Define the following parameters in the *Add Details* window that appears:
    - *Partitioning*: Enter an alphanumeric ID that is assigned to the partitioning
    - *Description*: Enter a description to name the partitioning
    - *Run Mode*: The run mode defines the system's behavior when a run for a function is triggered, or when a Model BW or Model Table with source environment is activated. The default run mode is specified in the partitioning setup.
      The following settings are available to determine the run mode:
        1. Parallel (P) or Sequential (S):
            1. "Parallel" means that the control returns immediately to the caller and does not wait for the function execution to be finished. The success of the execution is noted in the application log.
            2. "Sequential" means that the control returns to the caller only after the function execution is finished.
        2. Packaged (P) or Unpackaged (U):
            1. "Packaged" means that the ranges of the partitioning are used to trigger multiple instances of the function executions, each of them restricted to the field value defined in the range.
            2. "Unpackaged" means that one instance of the function execution is triggered without restriction to a range field value.
        3. Batch (B), Dialog (D) or Process like Caller (X):
            1. "Batch" means that a new background job is opened, the execution of the function is submitted to this background job and the job definition is closed afterwards.
            2. "Dialog" means that a new task is opened in dialog mode, where the execution of the function is triggered.
            3. "Process Like Caller" means that the execution of the function is triggered directly in the process of the caller (which can be either in dialog or background mode)
        4. Partitioned (P):
            1. "Partitioned" means that the environment managed Model Table or Model BW is activated in such a way that the partitioning range information is applied on the database. This is especially helpful in scale-out environments.
    - *Field*: Target field that is used as a partition key. You can set up the partition key in two steps:
        1. Register a field on the *Partitioning* tab within the environment. This field must be available in the input data being processed, which is then suitable for the logical separation of datasets into independent parts.
        2. Enter separate values for each partition to identify and select the data in the partition.

Example

A typical example is a version field, where the first partition is identified by the value “ACTUAL”, the second partition by the value “PLAN”, the third partition by the value “FORECAST”, and so on.

Example of Partitioning

Partitioning Field	VERSION
Partition Range	

Field Value	Level
ACTUAL	1
PLAN	2
FORECAST	2
SCENARIO 1	3
SCENARIO 2	4

You can define parallelization on top of a partitioning configuration by defining numeric level values for each partition value. By default, all partition values use the level value “1”, which means that all partitions are calculated in parallel during execution of level 1. If you change the level for single partition values, you can enforce sequential execution.

In the above example, 5 partition ranges for the field `VERSION` are set up. Based on the level, it is defined that the actual version is executed first, then the plan and forecast versions are executed in parallel on level 2. After that, scenario 1 is executed, and finally scenario 2.

Range Setup

Follow the steps below to set up the range:

1. Choose the *Add* button located in the *Range* section.
2. Define the following parameters in the *Add Details* window that appears:
   - *Range*: Enter an alphanumeric ID to which each package partition relates.
   - *Level*: This is used to control the number of packages to be executed in parallel or in sequence.
   - *Application Server Destination*: The name of the application server host, only applicable for the run modes “PPDP” (Parallel, Packaged, Dialog-Process, Partition) and “PPD” (Parallel, Packaged, Dialog-Process).

     ### Note

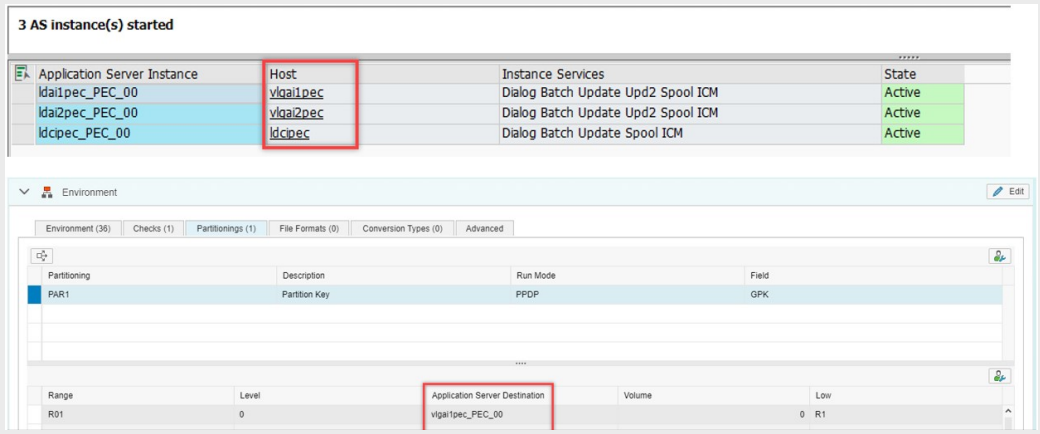
     By default (if you leave this field empty), a parallel-processed job in NetWeaver uses all qualified servers in an SAP system according to automatic resource allocation rules. In specific cases where you want to place the packaged calls over specific NetWeaver application servers, you can use this option.

     Follow the steps below to configure the application server destination:

     1. Enter transaction code `SM51` to get the destination name of the application server instance in the *Host* column.

2. Enter a value in the *Application Server Destination* field using the format **XXXXXXXX_SYS_IN**:
   - XXXXXXXX = Host name obtained from SM51
   - SYS = System name
   - IN = Instance

   Example

3. When you trigger the run, you can see in transaction SM50 which packages are being run in the specified application server instances, where the *Application Server Instance* reflects the name that you assigned in the *Application Server Destination*.

- *Volume ID*: In a scale-out system, the routing of a packaged procedure execution is usually handled implicitly by the SAP HANA database. That means, it decides which of the multiple hosts is used to execute the procedure.
  In some cases, you notice a particular host is loaded with a high number of procedure executions and you want to explicitly control the routing of the procedure execution to another specific host (for better distribution over multiple hosts to control memory and CPU resources). If you define a valid SAP HANA volume ID for a package, the SAP HANA HINT ROUTE_TO(<Volume ID>) is applied during the SAP HANA procedure call of that package fromSAP Profitability and Performance Management. SAP HANA carries out an explicit routing and tries to execute this package procedure on the host indicated by the volume ID. If an assigned volume ID does not exist in the system where it is being executed, the execution switches to the default behavior of implicit routing.
- *Low*: Single package selection values of the package or partition field

> **Note**
>
> For Model Tables, a change from non-partitioned to partitioned or vice versa updates the database immediately. For Model BWs, this change request is only recognized, and the BW administrator has to trigger the execution in the BW administration application.

1.2.5.2 Comparison of Different Run Modes

The run mode defines the system's behavior when a run of a function is triggered, respectively when a Model BW or Model Table with source environment is activated. The default run mode is specified in the partitioning setup (see Partitioning and Range Setup [page 21]).

If no partitioning is assigned to a function, a trigger for execution always uses the default run mode SUX, which has the settings “Sequential”, “Unpackaged” and “Process Like Caller”.

Example

The following example shows the system behavior depending on the different run mode settings:

Comparison of Different Run Modes

Partitioning Field	VERSION	
Partition Range	Field Value	Level
	ACTUAL	0
	PLAN	0
Run Mode	PPB (*Parallel, Packaged, Batch Process*)	

Partitioning Field	VERSION	
Partition Range	Field Value	Level
	ACTUAL	0
	PLAN	0
Run Mode	SUX (*Sequential, Unpackaged, Process Like Caller*)	

Sample Dataset

a) **Packaged**: System will package the dataset based on the partitioning field, in this case VERSION

VERSION	PRODUCT_ID	QUANTITY
ACTUAL	P001	40
ACTUAL	P002	20
ACTUAL	P003	10
ACTUAL	P004	50
ACTUAL	P005	30
PLAN	P001	60
PLAN	P002	20
PLAN	P003	50
PLAN	P004	80
PLAN	P005	10

b) **Parallel**: System will process the packaged dataset in parallel

VERSION	PRODUCT_ID	QUANTITY	
ACTUAL	P001	40	
ACTUAL	P002	20	
ACTUAL	P003	10	PROCESS 1
ACTUAL	P004	50	
ACTUAL	P005	30	
PLAN	P001	60	
PLAN	P002	20	
PLAN	P003	50	PROCESS 2
PLAN	P004	80	
PLAN	P005	10	

c) **Batch Process**: The processes will be assigned to the batch job (background job)

Note: You can see the actual process assigned in transaction code SM51

VERSION	PRODUCT_ID	QUANTITY		
ACTUAL	P001	40		
ACTUAL	P002	20		
ACTUAL	P003	10	PROCESS 1	BTC
ACTUAL	P004	50		
ACTUAL	P005	30		
PLAN	P001	60		
PLAN	P002	20		
PLAN	P003	50	PROCESS 2	BTC
PLAN	P004	80		
PLAN	P005	10		

Sample Dataset

a) **Unpackaged**: System will disregard the partition range and will not package the dataset

VERSION	PRODUCT_ID	QUANTITY
ACTUAL	P001	40
ACTUAL	P002	20
ACTUAL	P003	10
ACTUAL	P004	50
ACTUAL	P005	30
PLAN	P001	60
PLAN	P002	20
PLAN	P003	50
PLAN	P004	80
PLAN	P005	10

b) **Sequential**: System will process the unpackaged dataset by assigning one process (sequentially)

VERSION	PRODUCT_ID	QUANTITY	
ACTUAL	P001	40	
ACTUAL	P002	20	
ACTUAL	P003	10	
ACTUAL	P004	50	
ACTUAL	P005	30	PROCESS 1
PLAN	P001	60	
PLAN	P002	20	
PLAN	P003	50	
PLAN	P004	80	
PLAN	P005	10	

c) **Process Like Caller**: The processes will be assigned to either dialog job or batch job depending on the caller
Example: In the modeling UI, the process will be assigned to a dialog job, whereas in a CVPM process with „Run in Background, it will be assigned to a batch job

VERSION	PRODUCT_ID	QUANTITY		
ACTUAL	P001	40		
ACTUAL	P002	20		
ACTUAL	P003	10		
ACTUAL	P004	50		
ACTUAL	P005	30	PROCESS 1	BTC/DIA
PLAN	P001	60		
PLAN	P002	20		
PLAN	P003	50		
PLAN	P004	80		
PLAN	P005	10		

In the SUX example, the data records will be processed using one dialog or batch job sequentially

If you have assigned the partitioning ID to a function, it is used if this function is triggered for execution in the following applications:

- In the Modeling Environment

  > **i Note**
  >
  > You can overrule the standard described above in the *Advanced* tab of the run dialog.

- In the *My Activities* application
- In the *My Reports* application

1.2.6 Roles and Authorizations

SAP Profitability and Performance Management is targeted toward the business user. It is designed to enable the business department (for example, accounting, controlling and risk) to operate modeling, execution and analysis of data with minimal IT involvement. The solution is delivered with preconfigured user roles and provides each of them with a specialized working environment optimized to support them in their main area of responsibility.

The solution comes with the following predefined roles:

1. Administration Role `/NXI/P1_ADMIN_USER_ALL`
   Users assigned to this role can run the following transactions:
   - `Default Settings`
   - `Teams`
2. Modeling Role `/NXI/P1_MODELING_USER_ALL`
   Users assigned to this role can run the following transactions:
   - `Modeling Overview`
   - `My Environments`
3. Execution Role `/NXI/P1_EXECUTION_USER_ALL`
   Users assigned to this role can run the following transactions:
   - `Execution Overview`
   - `My Activities`
   - `My Events`
   - `My Reports`
4. Execution Management Role `/NXI/P1_EXECUTION_MANAGER` or `/NXI/P1_EXECUTION_MANAGER_ALL`
   Users assigned to this role can run the transaction `Processes`.
5. Management Role `/NXI/P1_SYSTEM_USER_ALL`
   Users assigned to this role can run the following transactions:
   - `Application Monitor`
   - `Process Monitor`
   - `Modeling History`
     By default, this role provides only display rights. To retrieve historic versions, the authorizations “Overwrite” and “Copy” are required. For more information, see below.

Granular Authorizations

In addition, you can maintain granular authorizations with the authorization object `/NXI/P1F` using the following fields:

1. `/NXI/P1ENV`
   This attribute defines the environment for which the authorization is maintained.
2. `/NXI/P1VER`
   This attribute defines the environment version for which the authorization is maintained.
3. `/NXI/P1PCU`
   This attribute defines the calculation unit for which the authorization is maintained.
4. `/NXI/P1FTY`
   This attribute defines the function type for which the authorization is maintained.
5. `/NXI/P1FID`
   This attribute defines the function ID for which the authorization is maintained.
6. `/NXI/P1ACT`
   This attribute defines for which action the authorization is maintained. The following values are allowed:
   - “Create”
   - “Display”
   - “Delete”
   - “Activate”
   - “Execute”
   - “Transport”
   - “Edit”
   - “Merge”
   - “Analysis”
   - “Remove”
   - “Copy”
   - “Overwrite”

You can use “*” as a placeholder for each authorization attribute to cover all the possible values of the attribute.

Related Information

For more information about SAP Profitability and Performance Management functions, see Functions [page 107].

1.2.7 Integration

1.2.7.1 Integration with SAP ERP and SAP S/4HANA

SAP Profitability and Performance Management allows integration with SAP ERP and SAP S/4HANA, including redundancy-free reuse of data, master data and hierarchies.

The solution uses SAP ERP and SAP S/4HANA capabilities to access accounting data in integrated scenarios:

1. Local Scenario
   In this scenario, the solution is installed on the same NetWeaver client. It directly uses the following:
   1. Reading of Master Data
      Master data attached to data elements and InfoObjects is reused.
   2. Reading of Hierarchy Data
      Hierarchy data attached to InfoObjects is reused.
   3. Reading Accounting and Controlling Data
      Accounting and controlling data available as SAP HANA-based CDS view interfaces is reused.
   4. Posting of Accounting and Controlling Data
      The official BAPI is used for posting via the `Remote Function Adapter`.
   5. Other use cases
      Further read or write access use cases can be customized using the `Model View`, `Model Table` and `Remote Function Adapter` functions.
2. Remote Scenario
   In this scenario, the solution is installed on a separate NetWeaver client or instance. It can remotely reuse the following:
   1. Reading of Master and Hierarchy Data
      Master and hierarchy data can be accessed remotely based on SAP HANA-based CDS view interfaces, but only during runtime.
      If this data needs to be accessed during design time in the modeling environment, the replication of the corresponding fields to local InfoObjects in the SAP Profitability and Performance Management instance has to be set up on the remote instance. For more information, see the SAP ERP and SAP S/4HANA documentation. Once this replication is set up, from an SAP Profitability and Performance Management perspective the master data and hierarchy data behaves as it does in the local scenario.
   2. Reading of Accounting and Controlling Data
      Accounting and controlling data available as SAP HANA-based CDS view interfaces from the remote SAP ERP and SAP S/4HANA instance can be reused.
   3. Posting of Accounting and Controlling Data
      For posting, the official BAPI is used. You need to specify the remote RFC destination on the *Advanced* tab for the environment.
   4. Other use cases
      Further read or write access use cases can be customized using the SAP Profitability and Performance Management functions `Model View`, `Model Table` and `Remote Function Adapter`.

Related Information

For more information about SAP Profitability and Performance Management functions, see Functions [page 107].

1.2.7.2 Integration with SAP Analytics Cloud and SAP Digital Boardroom

SAP Profitability and Performance Management allows easy integration with SAP Analytics Cloud and SAP Digital Boardroom. It also reuses the integration capabilities of BW and SAP HANA, and supports live data connections and import data connections with SAP Analytics Cloud.

SAP Analytics Cloud can access data using the following artifacts in integrated scenarios:

1. **Query Functions**
   SAP Analytics Cloud can access data using BW queries. The name of the BW query is visible in the function details header. This is the standard recommended design, because users can see in the solution exactly the same data as in SAP Analytics Cloud.
2. **Information Functions**
   SAP Analytics Cloud can also read data from various information functions . This is only necessary if the data needs to be processed further in SAP Analytics Cloud before the final results are presented to users.
   1. Model View
      Since model views do not hold data, but refer to a data source, SAP Analytics Cloud has to be configured to refer to the same data source as well.
   2. Model Table
      Access by SAP Analytics Cloud has to be configured according to the source type. The following options are available:
      1. Environment
         The data is managed by the solution and can be accessed via a SAP S/4HANA view. The name of the SAP S/4HANA view is displayed in the function header as the table name.
      2. All other source types
         All other model table source types refer to a data source. SAP Analytics Cloud has to be configured to refer to the same data source as well.
   3. Model BW
      Access by SAP Analytics Cloud has to be configured according to the source type. The following options are available:
      1. Environment
         The data is managed by the solution and can be accessed via a SAP S/4HANA view. The name of the SAP S/4HANA view is displayed in the function header as the view name.
      2. All other source types
         All other model BW source types refer to a data source. SAP Analytics Cloud has to be configured to refer to the same data source as well.
   4. Model Results Data
      Since model Results Data components do not hold data, but refer to a data source, SAP Analytics Cloud has to be configured to refer to the same data source as well.

Related Information

SAP Profitability and Performance Management Integration with SAP Analytics Cloud

1.2.7.3 Integration with BW, BPC and Analysis for Office

SAP Profitability and Performance Management allows the convenient integration with SAP Business Warehouse, SAP Business Planning and Consolidation and SAP Analysis for Microsoft Office, including redundancy-free reuse of data, master data and hierarchies.

SAP Business Warehouse Integration

The solution uses SAP Business Warehouse capabilities as an underlying Tool-BW in standalone scenarios. This includes relevant applications for management and maintenance of the BW and reusing Business Warehouse objects in integrated scenarios:

1. InfoObjects with master data and hierarchies
   The *Environment* function allows you to maintain managed InfoObjects and fields referring to BW managed InfoObjects.
2. Data Store Objects (Advanced)
   The *Model BW* function allows you to maintain managed ADSOs and to refer to BW managed DSOs and ADSOs.
3. InfoCubes
   The *Model BW* function allows you to refer to BW managed InfoCubes.
4. BW Queries
   The *Query* function allows you to maintain managed queries and to refer to BW managed queries.
5. Process Chains
   Process chains are generated automatically to control the vertical parallelization of functions involved in an activity.
6. Open ODS Views
   Open ODS views are generated automatically on top of nearly all functions to enable the analytic report screen.
7. Data Transfer Processes
   The Data Transfer Process can be used to store data in BW objects.
8. Characteristics-based Authorization
   BW characteristics-based authorization is used to secure and restrict access to data (for example, by legal entity or product group).

SAP Business Planning and Consolidation Integration

The solution reuses the following SAP Business Planning and Consolidation objects, including the relevant applications for managing SAP Business Planning and Consolidation objects:

1. Planning Application Kit (PAK)
   The *Writer* function allows you to hand results to the SAP HANA-based planning engine buffer. These results can then be reviewed by a user before deciding to save the data. For more information, see Analytics Component [page 76].
   The *Writer* function also allows you to store results using the SAP HANA-based planning engine directly in a BW Object. For more information, see Model Writer [page 323].

SAP Analysis for Microsoft Office Integration

Since SAP Profitability and Performance Management uses a lot of BW capabilities, it also uses all interfaces for a comprehensive SAP Analysis for Microsoft Office integration.

In the same way as in the web-based *Reporting & Simulation* application, it is possible to trigger writer functions of the BW write type "Planning" directly from SAP Analysis for Microsoft Office, and it is also possible to report and input data from Analysis for Office. For more information, see Analytics Component [page 76].

Related Information

For more information about SAP Profitability and Performance Management functions, see Functions [page 107].

For more information about InfoObjects, see Environment InfoObjects [page 54].

For more information about Data Store Objects (Advanced) and InfoCubes, see Model BW [page 114].

For more information about BW queries, see Query [page 347].

For more information about process chains, see Parallelization and Partitioning [page 20].

For more information about open ODS views, see Analytics Component [page 76].

For more information about HAP-based BW data transfer processes, see Model Writer [page 323].

For more information about characteristics-based authorization, see Roles and Authorizations [page 26].

1.3 Applications for Business Users

The following general applications are available for business users to help them streamline their work processes:

Administration

1. Export Environment [page 33]
   This application allows you to export an environment to a zipped .JSON file that you can store in a local directory.
2. Connection Management [page 34]
   This application allows you to manage and maintain connections to external systems.
3. Default Settings [page 39]
   This application allows you to enter specific settings for new environments, such as schema or path.
4. Teams [page 40]
   This application allows you to manage the teams that are the basis for assigning activities, events and reports to specific user groups.

Modeling

1. Modeling Overview [page 42]
   This application displays predefined statistics and KPIs relating to the usage of the modeling environment during design time.
2. My Environments [page 43]
   This application provides you with access to your modeling environments.

Execution

1. Execution Overview [page 68]
   This application displays predefined statistics and KPIs relating to the usage and behavior of the execution environment during runtime.
2. My Activities [page 69]
   This application provides you with a central access point for your processes and activities.
3. My Events [page 72]
   This application provides you with a central access point for your business events. You can also access any errors that may occur when processes and activities are executed. This application allows manual repairs as well as the configuration of automated situation handling rules.
4. My Reports [page 74]
   This application provides you with a central access point for your reports and what-if simulations.
5. Processes [page 93]
   This application allows key users to apply processes to user groups, including the setting of deadlines.

System Reports

1. Application Monitor [page 96]
   This application displays the detailed logs of all user and batch operations.
2. Process Monitor [page 98]
   This application provides an overview of the currently deployed processes.
3. Modeling History [page 98]
   This application displays the change history of all environments and allows users to retrieve historic versions.

Tools

1. Activate Function [page 101]
   This tool allows you to activate a function without having to navigate to the modeling environment.
2. Run Function [page 102]
   This tool provides support with running functions without having to navigate to the modeling environment.
3. Delete Temporary Data [page 106]
   This tool allows you to delete SAP Profitability and Performance Management Y tables, such as data produced by functions and the model table.

Related Information

See also Concepts for Key Users [page 6]

For more information about the available functions in SAP Profitability and Performance Management, see Modeling Environment [page 48].

1.3.1 Administration

1.3.1.1 Export Environment

This application allows you to export the complete configuration of an environment version into a zipped JSON file, which can then be placed in a local directory. You can use the downloaded file for convenient analyses or for subsequent processing activities. You can, for example, upload the file to the SAP Profitability and Performance Management Cloud system.

> **i** Note
>
> The environment to be exported must be in a consistent state. All functions which can be activated have to be successfully activated, and additional consistency requirements must be fulfilled. In case of inconsistencies, the system cancels the export and provides error messages containing information about the root cause.

You can start the *Export Environment* application using transaction code `/NXI/P1_FW_EXP_ENV`.

The necessary authorization is assigned to role `/NXI/P1_ADMIN_USER_ALL`.

Procedure

Follow the steps below to work with the *Export Environment* application:

1. In the client where SAP Profitability and Performance Management is installed, choose *SAP Menu* > *Profitability and Performance Management* > *Administration* > *Export Environment* or launch transaction code `/NXI/P1_FW_EXP_ENV`.

2. The system displays the *Export Environment* screen.
3. Provide the required information in the following fields:
   - *Env.+Version (EEEVVVV)*: Indicate the environment and version you want to export.

     > **i Note**
     >
     > You can use the F4 input help button to select the corresponding entries for the import in the list.

   - *Directory for Download*: Indicate a location in your local directory where the zipped JSON file shall be stored.

     > **i Note**
     >
     > You can use the F4 input help button to direct the path where the file shall be stored.
4. Choose *Execute* or the F8 button.

1.3.1.2 Connection Management

The *Connection Management* application is used for maintaining connections to external systems. It allows the logical mapping of external objects to be used for modeling in SAP Profitability and Performance Management.

Key Features

Key Features	Use
Check Connection	Performs the necessary checks to ensure that all relevant fields for the selected source type have been provided and that the external object is valid.
Connection is Obsolete	A connection which is already used by a model function cannot be deleted. Choosing this checkbox makes the connection unavailable for use. Such an obsolete connection is no longer available in the search help list of the model function's *Connection Name* field. Activating functions that still use the obsolete connection results in the error message “Connection is Obsolete”.

Key Features	Use
Upload from Excel	Allows you to upload a connection setting or multiple connection settings. **i Note** A connection with an existing name and class is updated if the checkbox *Overwrite Existing Data* in the file upload screen is selected. If it is not selected, the existing connections are not updated and only the new connection settings are added. For more information about the file upload template in *Connection Management* of SAP Profitability and Performance Management, see SAP Note 3038623.

<table>
<tr><th>Key Features</th><th>Use</th></tr>
<tr><td>Transport</td><td>

Transport Preparation in the Source System

Upon environment transport, the “Default” connections used by an environment from the source system is considered and transported to the target system.

In contrast to the "Default" connection class, the system cannot transport “Custom” connections from one environment to another system as "Custom" connections are only available locally.

> **i** Note
>
> If the target system does not allow for Connection Management customization, make sure to set up the "Default" connection class that needs to be transported with the required settings of the target system.

Transport’s After-Import Activation Phase in the Target System

A system can use the same connection name with different connection classes.

Example:

Connection Name	Connection Class	Pointing to
CON1	DEFAULT	DB1
CON1	CUSTOM	DB2

If a "Custom" connection (in the example above pointing to DB2) exists in the target system, the system uses this connection during activation, whereas the "Default" connection is ignored.

However, if no "Custom" connection with the same connection name exists in the target system used by the environment, the "Default" connection included in the transport or available in Connection Manager is used during activation.

</td></tr>
<tr><td>Migration Report</td><td>

This report allows you to migrate all functions from all environments in a single report run. The report also creates required connections. It can be executed as many times as required without consequences, given that only the first execution changes the data.

For migrating model functions from physical to logical models that use connections, launch the report `/NXI/P1_MIG_MODEL_PHY_TO_LOG`.

</td></tr>
</table>

Procedures

Adding a Connection

Follow the steps below to add a new connection:

1. In the client where SAP Profitability and Performance Management is installed, choose *SAP Menu* > *Profitability and Performance Management* > *Administration* > *Manage Connections* or launch the transaction code `/NXI/P1/CONM`. The system opens a browser window where you can add connections in edit mode.
2. The *Connection Management* screen appears. Choose the *Edit* button in the top left-hand corner of the screen to continue.
3. Choose *Add*.
4. In the *Add Details* window, fill out the following available fields:
   - *Connection Name*: Alphanumeric and unique name with a maximum length of 30 characters

     > **i Note**
     >
     > Connection names starting with "S" are reserved for the sample content. Therefore, make sure that the connection name you create does not start with the letter "S".

   - *Connection Description*: Details about the connection
   - *Connection Class*
     - "Default": This connection can be transported
     - "Custom": This connection class cannot be transported
   - *Connection Source*: Connections are allowed for DDIC Table, DDIC View, CDS View, HANA Table, HANA View, HANA SDA, BW InfoProvider, Model RDL and OData Service.
5. Choose *OK* to continue.
6. The system displays the *Connection Details* screen, where you can see the *Connection Name*, *Connection Details*, *Connection Source* and *Connection Description* columns. On the left-hand side, these field columns are read-only; they can be edited only on the right-hand side. You can add hidden columns by using the *Personalization* button. Additional fields are available in the connection details on the right side.
7. In the *Table / View Name* field, you can choose a table name either by entering it manually or by using the search help as follows:
   1. Choose the *Search Help* button.
   2. The *Select: Table Name* dialog box appears. Make an entry in the *Table Name* field and press the `Enter` key.
   3. Select an entry in the results section to automatically transfer it to the *Table Name* field.

   The following fields are important for setting up the OData service connection:
   - *RFC Destination*: When you use the OData service from an external system, the HTTP RFC destination is required from source to target system.
   - *Decimal Separator*: You can use a comma or period which serves as the reference for data conversion.
   - *Service Base URL* (mandatory): Identifies the root of the OData service.
   - *EntitySet Name* (mandatory): Identifies the resource which is involved in the interaction.
   - *Service URL Options* (optional): Use this field to define common query options, for example select and filter.
   - *Maximum Page Size* (optional): Specifies if the request should be executed using a maximum page size. If you leave this field blank, the server-side maximum page size is used if it exists.

- *Authentication Type* (mandatory): Specifies which authentication type the service uses.
  - If no authentication is required, select the "Basic Authentication" option. In this case, the OData service connection is established within an `SM59` connection.
  - If the service shall provide an authentication, select the "OAuth"option. In this case, only the authentication request is processed using an `SM59` connection. The URL of the actual service must be maintained.
- *Header Fields* (optional): Some OData services only work with additional header fields. You can maintain such fields here using the `FIELD1=VALUE1;FIELD2=VALUE2` format.

> **i Note**
>
> You can only use the OData Service in a *Model Table* function, and only for reading data. To populate the Ytable of the function that consumes the OData, you must execute the function via the *Run* tool. For more information about running functions, see Run Function [page 102].

8. Save your changes.

Removing a Connection

A connection can be removed only if it is not used by any functions.

> **i Note**
>
> If the connection has both a default and custom class, you are allowed to delete one of them even if the connection is used by a SAP Profitability and Performance Management function.

Follow the steps below to remove a connection:

1. In the client where SAP Profitability and Performance Management is installed, choose *SAP Menu* > *Profitability and Performance Management* > *Administration* > *Manage Connections* or launch the transaction code `/NXI/P1/CONM`. The system opens a browser window where you can delete connections in edit mode.
2. On the *Connection Management* screen, choose the *Edit* button in the top lefthand corner to continue.
3. Choose the connection to be deleted, and then choose *Delete*.
4. Confirm the deletion by choosing "Yes" in the *Confirm Deletion* dialog box.
5. If the connection is used by a function, the *Where Used* dialog box appears. It shows a list of functions from all environment versions that use the selected connection.
6. To continue with the deletion, go to the modeling environment and disassociate the connection from the function. Save your changes and perform steps 1 and 2 again.

Managing Connections in a Model Function

The following connection types are available for the model function types *Model Table*, *Model View*, *Model BW* and *Model RDL*:

- "Physical": The function is configured normally.
- "Logical": The function needs only the field *Connection Name*; all other fields are not required.

Follow the steps below to use a connection in a model function in the environment:

1. In edit mode, choose the *Add* button.
2. Fill in the required fields:
   - Function

- Description
- Function Type

3. Fill in any optional fields as necessary:
   - Event Handling
   - Logging
   - Processing Type
   - Partitioning
4. Choose *Ok* to save your changes.
5. In edit mode, fill in the following function details:
   - Connection Type
     - Physical

       > **i Note**
       >
       > If you choose this option, follow the procedures on how to create one of the following model functions:
       > - Model Table [page 134]
       > - Model View [page 137]
       > - Model BW [page 114]
       > - Model RDL [page 131]

     - Logical
   - Connection Name
6. Choose the *Synchronize* button. The fields associated with the connection name are listed.
7. Activate the function.

1.3.1.3 Default Settings

Default settings are applied to every new environment.

The default settings are defined in a default environment with the name “Environment Template for Default Settings” and the ID “SAP” with version “1”. When a new environment is created, the default environment setting or configuration is copied to the new environment.

Typical default settings include the environment database connection. Other settings, such as fields or functions, can be defined in the default settings and are also copied to every new environment.

You can use this function if your origanization has a set of formats that needs to be followed or considered. If you change or add to the default settings, your required settings and formatting standards will then automatically be applied when modelers create a new environment.

Procedure

1. In the client where SAP Profitability and Performance Management is installed, choose *SAP Menu* > *Profitability and Performance Management* > *Administration* > *Manage Default Settings*.

> **Note**
>
> The system displays the same environment as the one defined in *SAP Menu* > *Profitability and Performance Management* > *Modeling* > *Start My Environments* > *Environment Template for Default Settings (Environment SAP, version 1)*.

2. In *Edit* mode, you can add nodes, functions, fields, and settings that you want to be considered a "template".
3. Once you have finished configuring your template, choose *Save*.

> **Note**
>
> Everything that you add here and save will then be part of all new environments that are created in the same system.

Related Information

For more information about modeling entities, see Financial and Business Modeling Entities [page 7].

For more information about the available functions, see Functions [page 107].

1.3.1.4 Teams

Teams are groups of users that work together on processes, business events, and reports. Multiple teams are typically used in decentralized processes, where different activities have to be executed by different groups of users. Each user can see only those activities that are assigned to their team.

Application managers can control the teams and the assignment of users to a team.

Key Features

The following key features are available:

Key Feature	Use
Team Management	Teams (user groups) can be created, edited, and removed. Teams are available across all environments and even for other applications in the same client, such as SAP Business Workflow. Specific users can be added to a team and removed from a team. Users are created, edited, and deleted centrally by SAP NetWeaver administrators and not in this application.
Assignment of Teams	Teams can be assigned to activities during the deployment of processes, so that these activities can be performed by the users that belong to that team.

Procedure

1. In the client where SAP Profitability and Performance Management is installed, choose *SAP Menu* > *Profitability and Performance Management* > *Administration* > *Manage Teams*.
2. The system opens a browser window that has two screen sections:
   Left section: team creation window
   1. Choose *Add* and enter a name and description to create a team.
   2. Choose *Remove* to remove or delete any existing teams. If the team is being used in workflows or processes by SAP Profitability and Performance Management, the system does not allow you to delete them.

      > **i Note**
      >
      > If the team is being used by process activities, the system displays a *Where-used* screen.
      >
      > An administrator can only delete a team after removing the assignment in all process activities shown in the *Where-used* list.

   3. If you want to change the description, choose *Edit* and rename the description of the team.

   Right section: assigning users to teams window

   > **i Note**
   >
   > Users must have already been created. This must be done by user administrators.

   1. On the left-hand side of the screen, choose the team for which you want to add or remove a user.
   2. Choose *Add* to add a user to the team that you have selected.

3. Choose *Remove* to remove a user from the team you have selected.

Related Information

For more information about the use of teams in processes, see Manage and Deploy Processes [page 93].

1.3.2 Modeling

1.3.2.1 Modeling Overview

The modeling overview displays various key performance indicators that are relevant for modeling, such as biggest and smallest environments and activation times.

Key Features

The following key features are available:

Key Feature	Use
Modeling Key Performance Indicators	The modeling overview provides an overview of all your environments, including their size, frequency of change, and other useful information. It also informs you whether function templates for reuse are available.
Key Performance Indicator Graphs	All key performance indicators are displayed in a graphical format. Each graphic is interactive. You can navigate from the elements to the corresponding environment, the application monitor, the process monitor, and the modeling history. Graphics can also be displayed in a tabular format.

> **i Note**
>
> To use the full functionality of *Modeling Overview*, we highly recommend to launch SAP Profitability and Performance Management via Fiori Launchpad.

Related Information

For more information about financial and business modeling, see Financial and Business Modeling Entities [page 7].

For more information about the modeling environment, see Modeling Environment [page 48].

1.3.2.2 My Environments

An environment is a versioned group of shared metadata, functions, and information that comprises a financial and business model. It can be managed in the system landscape without affecting other environments.

You can define multiple environments, and can use nodes to structure environments for different purposes.

Key Features

The following key features are available:

Key Feature	Use
Environment Management	Environments and their versions can be added, edited, copied, merged, removed, and transported. Changes made in an environment are not only saved but also historized with information about who made the change and when. This information is available in the *Modeling History* application. For more information see Modeling History [page 98].
Nodes Management	Nodes can be used to structure multiple environments in a hierarchy and, like directories, can contain other nodes. Nodes can be created, edited, removed, and transported but, unlike environments, do not have a version.
Authorizations	Authorizations can be attached to environments and nodes so that they can be viewed or edited only by selected modeling users. While SAP Profitability and Performance Management provides a dedicated application for team management, authorizations are managed centrally by the SAP system administrator.
Modeling Environment	When you select an environment, the system opens the modeling environment application where you can use the *Continue* button to maintain the environment.

Procedure

In the client where SAP Profitability and Performance Management is installed, choose *SAP Menu* > *Profitability and Performance Management* > *Modeling* > *Start My Environments*. The system opens a browser window where you can process the following activities in *Edit* mode.

Add

You can use the *Add* button to create a new node or environment:

1. Choose an entry from *My Environment* to be the starting point of the node or environment that you are creating.

   > **i Note**
   >
   > If you don't choose a starting point, the environment or node you create will be added at the first level in the structure.

2. Choose + (*Add*) to open the *Add Environment* window.
3. Make entries in all of the fields as follows:
   1. **Add Level**
      You can choose one of the following options:
      - "Same Level": This option means that the environment or node that you create is structured or created at the same level as the selected entry. If you have not selected an entry before choosing + (*Add*), the system automatically creates the environment or node at the highest level of the structure .
      - "One Level Below": You can only use this option if you have selected a node before choosing + (*Add*). You can only add nodes or environments to nodes or directories.

      If you selected an environment before choosing + (*Add*), the system automatically sets the *Add Level* field to "Same Level".
   2. **Environment ID**
      This is a 3-digit alphanumeric ID that is permanently assigned to the node or environment. Once it has been created, it is not possible to edit the environment ID.

      > **i Note**
      >
      > In SAP Profitability and Performance Management, do not use an environment ID that starts with the letter "S" (for example "SEN"). This letter is reserved exclusively for use in the default template environment and for sample content included in SAP Profitability and Performance Management releases and support packages.

   3. **Environment Type**
      You can choose one of the following options:
      - "Environment Version": You can use this option if you intend to create an environment where configuration can take place
      - "Node": This option is used to structure the environments and entries on the *My Environment* screen.
   4. **Use Default Settings**
      You can choose one of the following options:
      - "Yes": If you select this option, you can reuse and adapt the configuration made in the *Default Settings Environment* (SAP version 1)

- "No": If you select this option, the system creates an environment that is completely blank and you need to adjust the full environment, including the assignment of the database connection.

5. **Version**
   This is only a required field for environments (not for nodes). The version is a 4-digit alphanumeric ID that is permanently assigned to the environment. Once the system has created it, it is not possible to edit it.
6. **Description**
   Enter a description here to name the environment or node. You can adjust this description later, if necessary, by choosing *Edit* on the *My Environment* screen.

4. Choose *Save* for the changes to take effect.

Edit

You can use the *Edit* button to change the description of the node or environment:

1. Select the node or environment that you want to edit.
2. Choose *Edit*.
3. A popup window appears where you can change the description.
4. Choose *OK*.
5. Choose *Save* for the changes to take effect.

Copy

You can use the *Copy* button to copy an environment (never a node). The system automatically copies all the configuration carried out in the environment . It also activates some functions, such as *Model Table* with the option "Transport Yes" or functions that are added as activities of a process template.

Follow these steps:

1. Choose an environment that you want to copy.
2. Choose *Copy*.
3. The system displays a popup window where you need to provide the following information:
   1. **Environment ID**
      This is a 3-digit alphanumeric ID that is permanently assigned to the environment. Once the system has created it, you can no longer edit it.

      > **i** Note
      >
      > In SAP Profitability and Performance Management, do not use an environment ID that starts with the letter "S" (for example "SEN"). This letter is reserved explicitly for use in the default template environment and for sample content included in SAP Profitability and Performance Management releases and support packages.

   2. **Version**
      The version is a 4-digit alphanumeric ID that is permanently assigned to the environment. Once the system has created it, you can no longer edit it.
   3. **Description**
      Enter a description here to name the environment or node. You can change this later, if necessary, by choosing *Edit* on the *My Environment* screen.
4. Choose *OK*.
5. Choose *Save* for the changes to take effect.

Merge

You can use the *Merge* button to merge the configuration of one environment with the configuration of another environment (never a node).

The merge function not only adds or merges functions from one environment to another, it also merges functions, formulas, and rule types for example.

> **Example**
>
> If a function called “ZPER” exists in both Environment 1 and in Environment 2, all configuration settings for both ZPER functions are merged, including the formula and rules used.

Follow the steps below:

1. Choose an environment. This is the environment that will be merged with the other environment, meaning all configuration settings from this environment will be transferred to the other environment.
2. Choose *Merge* .
3. The system displays a popup window. Select the environment that you want to merge the environment from step 1 with.
4. Choose *OK*.
5. The system copies or merges all functions and rules from the environment of step 1 to the environment from step 3.
6. Choose *Save* for the changes to take effect.

Remove

You can use the *Remove* button to remove an environment and/or node, and then transport this cleanup process or deletion to other systems using a transport request.

Follow the steps below:

1. Choose the environment or node that you want to delete.
2. Choose *Remove* .
3. The system displays a popup window where you can enter the following information:
    1. Comment section
       You can use this optional field to document why deletion is necessary, for example.
    2. Transport request selection
        - “Available Transport Request”: Displays all Customizing requests assigned to the user deleting an environment.
        - “Create New Transport Request”: Creates a new Customizing transport for the user deleting an environment.
        - “No Transport”: Does not save the changes to a transport. This means the same environment will not automatically be deleted in the next system.
4. Choose *Confirm* to automatically remove the environment or node (plus everything underneath it). In this case, you do not need to choose *Save* for the changes to take effect.

Transport

To transport an environment from one client to another, or from one system to another, SAP Profitability and Performance Management uses the standard transport management provided by SAP.

An environment that needs to be transported is saved to a Customizing transport that can then be imported from one system to another. This means that all configuration settings in the environment are copied automatically to another system by means of TMS (Transport Management System).

To transport an environment, follow the steps below:

1. Select an environment or set of environments that need to be transported.

> **i Note**
>
> Do not select the node since it will automatically be included in the transport. Select only the environment when you create a transport.

2. Choose *Transport*.
3. A popup window with the following options appears:
   - "Available Transport Request": This displays all Customizing requests assigned to the user.
   - "Create New Transport Request": This option creates a new Customizing transport for the user.
4. Choose *OK* to automatically add the environment settings and configuration settings to the transport. This can then be imported to the next system once it is released.

> **i Note**
>
> - For the development system (DEV), the settings should not cause a conflict with the number range for the quality system (QA) and the productive system (PRD).
>
>   **Example**
>
> | System | From | To |
> |---|---|---|
> | DEV | 000000 | 199999 |
> | QA | 200000 | 399999 |
> | PROD | 400000 | 599999 |
>
> - The RFC connection user must be assigned the authorization `/NXI/P1_MODELING_USER_ALL` since it will generate the procedure during transport.
>   For more information, see "Create RFC Destination" in the Administration Guide for SAP Profitability and Performance Management.

The system activates some functions after transport. See the *Transport* section in the Administration Guide for SAP Profitability and Performance Management for more details.

The system also transports all the contents of the specific tables to the environments selected. To check all the tables included in the transports, follow the steps below:

1. In the SAP NetWeaver system, launch transaction code `/n/nxi/p1_mf`.
2. Choose *Function Table Info*.

> **i Note**
>
> The *Function Table Info* overview screen appears. The listed table names are included once a transport is performed in the Modeling Environment.

> If you are using SAP Profitability and Performance Management 3.0 SP07 or below, select *Meta Function Table Info*.

3. The *Function Table Info* overview screen appears. The listed table names are included once a transport is performed in the Modeling Environment.

Related Information

For more information about the modeling environment, see Modeling Environment [page 48].

1.3.2.2.1 Modeling Environment

You can use the the modeling environment to set up and change financial and business models. This is where all model designs, changes and enhancements are made to meet the requirements of specific use cases.

You can set up a model from scratch or based on a copy of one of the sample content models that you can then adjust to meet your specific needs.

1.3.2.2.1.1 Screen Header Buttons

The following buttons are available in the header of the *Modeling Environment* screen:

- **Save**
  Records all the changes made in the environment to ensure that you do not lose all the configuration settings you have made.
  There is an automatic save mechanism when you do not save the changes made in the environment and move to another section of the environment.

  > **i** Note
  >
  > In newer versions of SAP UI the *Save* button is in the lower right corner of the screen.

- **Show/Hide**
  You can use the *Show/Hide* button in the screen header to collapse and expand the function hierarchy, which gives you a better visualization of the function details during configuration.
- **Environment Details [page 49]**
  Opens the details of the environment, where the system displays six sections that contain settings that apply to specific functions in the environment.
- **Historize**
  Takes a snapshot of the current saved status of the whole environment configuration, including all field and function details, and saves this snapshot in the modeling history.

> **Note**
>
> We recommend you do this before making bigger changes to an environment because it allows you to restore the snapshot later, if needed.

- **Modeling Flow**
  Shows a visual representation of an environment version in the form of a directed graph, where you can see the dependencies between business functions.
  To know more about the modeling flow, refer to the Studio [page 77] section.

1.3.2.2.1.1.1 Environment Details

The *Environment* button in the screen header opens the details of the environment. There are six sections which contain settings that are valid for specific functions in the environment:

- Environment [page 49]
- Checks [page 58]
- Partitioning [page 60]
- File Formats [page 60]
- Conversion Types [page 62]
- Advanced [page 63]

1.3.2.2.1.1.1.1 Environment

On this tab you can define internal and external fields such as the following:

- All Fields [page 53]
- Environment InfoObjects [page 54]
- Environment Fields [page 55]
- BW Fields [page 57]
- DDIC Fields [page 57]
- HANA Fields [page 58]

To edit fields in this section, please take note of the available functions (see Buttons [page 49]) and the field attribute definitions (see Field Attributes [page 51]).

1.3.2.2.1.1.1.1.1 Buttons

You can use the following buttons when you work with environments:

- **Add**
  Adds the field to be used in the modeling environment.

- **Remove**
  Deletes a field in the environment if it is no longer relevant to the environment.
  To remove a field, follow the steps below:
  1. Select the field to be deleted and choose *Remove*.
  2. A confirm deletion window appears, where you can add a comment if required..
  3. Choose *Confirm*.

  > **i** Note
  >
  > You cannot remove a field if it is being used by a function.

- **Copy**
  Duplicates the field and all of the properties of the field.
  To copy the fields, follow the steps below:
  1. Select the field to be copied and choose *Copy*.
  2. A copy detail window appears.
     1. Choose the position to whichyou want to copy the field. You can select from the following two options
        - **First Row**: The field will be placed in the first row
        - **Below Row Number**: The field will be placed below the row of the selected field.
     2. In the table below the target position, the system displays the following information:
        - *InfoObject*: Field, Description and InfoObject
        - *Environment InfoObject*: Field and Description

        The column on the left is the source field; the column on the right is the target field.
     3. Define in the column on the right side where to copy the field to (in other words, the copy target).
     4. Choose *Ok*.
- **Open Master Data**
  If you need to define the master data for a field, select it to enable the *Open Master Data* button.

  > **i** Note
  >
  > On the *Environment Fields* tab this button is displayed as *Open Master & Hierarchy Data*.

  To create the master data, follow the steps below:
  1. Choose *Open Master Data*.
  2. A new window appears, where you can define the master data for the selected field.
  3. Add a line and list the data.
  4. Once you have completed your entries, choose *Save*.

  > **i** Note
  >
  > Only fields of type “Characteristics” can have master data.

- **Open Hierarchy**
  If you need to define a hierarchy for a field, select it to enable the *Open Hierarchy* button.

  > **i** Note
  >
  > On the *Environment Fields* tab this button is displayed as *Open Master & Hierarchy Data*.

  To create a hierarchy, follow the steps below:

1. Choose *Open Hierarchies* .
2. A new window appears, where you can define the hierarchy for the selected field.
3. Choose *Create Hierarchy* and fill out all the necessary information.
4. Once you have completed your entry, choose *Save* and activate.

- **Where used**
  Lists all the functions that use a selected field.
  To check where the fields are used, follow the steps below:
  1. Select the field and choose the *Where used* button.
  2. A dialog box appears, showing all the functions that use the field.

> **i** Note
>
> When you click on a function in this overview, you can directly open the function itself.

1.3.2.2.1.1.1.1.2 Field Attributes

Available Field Attributes

Field Attribute	Description
Aggregation	This field attribute only applies to fields of the type “Key Figure”. There are three aggregation options: • **Minimum**: The minimum value of all the values in this column is displayed in the results row. • **Maximum**: The maximum value of all the values in this column is displayed in the results row. • **Summation**: The total of all the values in this column is displayed in the results row.
BW Data Type	Specifies which BW data type is used for the field being created, such as “Character String” or “Packed Numbers in BDC Format”.
Class	There are two options to create entries in the *Environment* field: • **Field** can be used in all functions to work on data. A typical example of a field is a financial period, which identifies for which month the data is valid. • **Parameters** are used to steer processes and calculations. They are used in formulas and calls of certain functions to influence the logic and operations applied.
Conversion Routine	This field attribute allows the characteristic values (key) of a field to be displayed or used in a different format from the format that they are stored in. The most commonly used conversion routine is “Alpha”.

Field Attribute	Description
Data Decimal	You need to define this field attribute if the field is of type "Key Figure" and requires a decimal notation, such as "DEC - Packed Number in BCD Format" and "CURR - Currency Field in BCD Format". **i Note** The maximum length that can be used for decimal is 15.
Data Length	Defines the length of a value in a field.
DDIC Type	Specifies which DDIC type is used for the field being created, such as "CHARS", "DATS" or "DEC - Packed Number in BCD Format".
Description	*Description* is a business-like name of the field being created. It has a predefined characteristic value that you can use as it is or change.
Field	*Field* is a unique technical name that initially has a predefined characteristic value which you can use as it is or change to a different value.
Field Type	Contains three options that allow you to select which field type is used for the field being created: • **Characteristic**: Used to identify key figures, and contains texts, codes, dates or numerical characteristic values. • **Key Figure**: Used for calculations and can contain natural numbers, integers, decimals or floating points. • **Unit**: Required to give meaning to the values of a key figure. **i Note** Key figures of the type "Currency" are always assigned a currency key, and key figures of the type "Quantity" are also assigned a unit of measurement.
Hierarchy	You select this field attribute to define a hierarchy for a field.
Hierarchy Assignment	You can select the following options to be used with the result row output from the *Query* function: • **Minimum**: The minimum value of all the values in this column is displayed in the results row. • **Maximum**: The maximum value of all the values in this column is displayed in the results row. • **Summation**: The total of all the values in this column is displayed in the results row.

Field Attribute	Description
Hierarchy Date	If you want to create a hierarchy that is valid for a specific time period, you need to create the entire hierarchy as time-dependent.
Hierarchy Name	You can define the created hierarchy of the field, so that the modeler can use the specific hierarchy name when structuring the data in a query.
Hierarchy Version	The modeler can define the created hierarchy of the field so that the modeler can use the specific version of the hierarchy when structuring the data in a Query.
InfoObject	InfoObjects are the basic information providers of SAP BW. They structure the information needed to create data targets such as infocubes and master data. They can be classified into characteristics, key figures, units, time characteristics and technical characteristics.
Item Unit Assignment	You can uset this field attribute o define a fixed unit or currency. You can set up either Unit Field or Item Unit assignment to define a unit of the key figure.
Key Figure Type	Specifies which type of key figure is used for the field being created. **i Note** Amounts and quantities need unit fields.
Lower Case	You mark this checkbox when you want lower case entries to be accepted for data records. By default, this field attribute will follow the properties set for reference InfoObjects.
Master Data	You select this field attribute to define the master data of a field.
Reference InfoObjects	You can opt to use the attributes of an existing InfoObject in the system so that the field can share the properties and metadata of that InfoObject with the field that is being created.
Unit Fields	If the key figure created is of type “Currency” or “Quantity”, you should define the unit field. This assigns the unit field to a key figure field. If the key figure is of type “Currency” or “Quantity”, the drop-down list is enabled for the unit field column and displays the list of possible unit fields to be assigned for that key figure.
Unit Type	For unit key or currency key figures, you need to define this field attribute to enter a fixed currency or unit.

1.3.2.2.1.1.1.1.3 All Fields

This section shows a summary of the fields that are available on each field tab.

Consistency Check

Validates the correctness of all the fields available in the environment.

To perform the consistency check, follow these steps:

1. On the *All Fields* tab, choose *Consistency Check*.
2. The system displays a message indicating that no inconsistencies have been found in the environment.

1.3.2.2.1.1.1.4 Environment InfoObjects

Fields can refer to other InfoObjects that share metadata. Once the environment is transported, the objects created for this field are also generated in the target system. For more information, see Field Attributes [page 51] .

Adding Fields

To add a field, follow the steps below:

1. Choose the *Add* button.
2. The *Add Details* dialog box appears. Here, you need to define the following: *Field*, *Description*, *Field Type*, *Reference InfoObject* and *InfoObject*. For more information about the attributes, see the Field Attributes [page 51] section.

   > **i Note**
   >
   > *InfoObject* is a 9-digit alpha-numeric field that must start with a “Y”. The system uses the value that you enter in this section as the name of the generated BW InfoObject.

3. Choose *Ok*.
4. Fill in additional attributes depending on the selected field type and the chosen *Reference InfoObject* settings.
   If the *Reference InfoObject* is not defined, you have to specify the following properties for:
   - **Characteristic**
     Make entries for *BW Data Type Key* and *Data Length*.
   - **Key Figure**
     Make entries for *BW Data Type*, *Data Length*, *Data Decimal* and *Key Figure Type*.
     If a key figure is a currency or quantity, you can set up either a unit field or an item unit assignment to define a unit for the key figure.
   - **Unit**
     Make entries for *BW Data Type* and *Unit Type*.

   If the *Reference InfoObject* is defined, the different properties of the field (such as *BW Data Type*, *Data Length*, *Conversion Routine*, *Master Data* and *Hierarchy*) are automatically filled. The system adapts these properties from the reference InfoObject to the created environment InfoObject.
5. You still have the option of defining the *Conversion Routine*, *Master Data* and *Hierarchy*. For more information, see the procedure for *Master Data* and *Hierarchy* in the Buttons [page 49] section.

6. Choose *Save*.

Removing or Copying Fields

To remove or copy a field, follow the *Remove* or *Copy* procedure in the Buttons [page 49] section.

Where used

To find out where a field was used within a function, follow the *Where used* procedure in the Buttons [page 49] section.

1.3.2.2.1.1.1.5 Environment Fields

Fields are visible only in the environment in which the field is defined. Once the environment is transported, these fields will also be available in the target system. Refer to the Field Attributes [page 51] section for details.

Adding Characteristics Field

To add a field, follow these steps:

1. Choose *Add*.
2. The *Add Details* dialog box appears, where you need to define the following field attributes: *Field*, *Description*, *Field Type*, *Class* and *DDIC Type*.

   > **i** Note
   >
   > The DDIC type is dependent on what field type has been selected.

3. Choose *OK*.
4. On the *Environment Field* tab, define the *Data Length*.
5. You can also define, master data and a hierarchy, define attributes, define lower case and a conversion routine.

Open Master and Hierarchy

To define a master data or a hierarchy for a field, select the *Master Data* or *Hierarchy* checkbox.

Both *Open Master & Hierarchy Data* and *Maintain Attributes* buttons will be enabled. Only fields from the "Characteristics" type can have master data.

To create master data and/or a hierarchy, do the following:

1. Choose *Open Master Data & Hierarchies*.
2. The data editor window will appear. Choose *Edit* to define the master data and a hierarchy.
   To define master data, follow these steps:
   1. Specify the *Value*, normally a text which will be included in the list of master data.
   2. Specify the *Description* to describe the value you have entered.

   To define a hierarchy, follow these steps:
   1. Select the *Is Node* checkbox to enter a node or a parent of the structure from the master data list.
   2. Specify the *Parent Value* to assign the master data to a specific node.
   3. Specify the name of the hierarchy.
   4. Choose *Save*.

Maintain Attributes

You can opt to use the master data of a different field that is created within the *Environment Fields* tab.

To use the master data of another field, follow these steps:

1. Choose *Maintain Attributes*.
2. The *Maintain Fields* dialog box will appear, where you can choose the fields from the list.
3. To choose a field, click on the checkbox.
4. Choose *OK* to continue.
5. Choose *Save*.

Adding Key Figure or Unit Field

To add a field, follow the steps below:

1. Choose the *Add* button.
2. The *Add Details* dialog box will appear, where you need to define the following: *Field*, *Description*, *Field Type*, *Class* and *DDIC Type*.

   > **i Note**
   >
   > The DDIC type is dependent on the selected field type. A field from the "Unit" type can be either "CUKY" for a currency field or "Unit" for a quantity field.

3. Choose *OK*.
4. On the *Environment Field* tab, define the *Data Length*. You have the option of defining a decimal data type for key figures.
5. Choose *Save*.

> **i Note**
>
> If you need to create a parameter field, select "Parameter" as the *Class*.

Removing or Copying Fields

To remove or copy a field, follow the *Remove* or *Copy* procedure in the Buttons [page 49] section.

Where used

To find out where a field was used within a function, follow the *Where used* procedure in the Buttons [page 49] section.

1.3.2.2.1.1.1.6 BW Fields

These are similar to BW InfoObjects but no master data or hierarchies are available in the source for these fields.

The fields are added in the environment if an InfoProvider for a *BW* field has been consumed in one of the information functions.

1.3.2.2.1.1.1.7 DDIC Fields

Data types and master data are maintained by an external application and are used in the environment as part of the model. This means that the fields are registered in the environment and refer to their original source.

Adding Fields

Add the field you want to use in the modeling environment. Follow the steps below:

1. Choose the *Add* button.
2. On the *Add DDIC Fields* dialog box you can specify the *DDIC Data Element* as a search criterion. Choose *Search* to find the field. Optionally, you may indicate an asterisk on the *DDIC Data Element*, so that the search will provide the full list of DDIC fields.
3. To add the field(s) to your environment, select the field(s) in the generated lists.
4. Choose the *OK* button and then *Save*.

Removing or Copying Fields

To remove or copy a field, follow the *Remove* or *Copy* procedure in the Buttons [page 49] section.

Where used

To find out where a field was used within a function, follow the *Where used* procedure in the Buttons [page 49] section.

1.3.2.2.1.1.1.8 HANA Fields

To add fields in this tab, consume an InfoProvider for a HANA table for HANA fields.

Alternatively, you can add a field on this tab using the mapping functionality from the *File Adapter* function.

When you use the field mapping proposal in the file adapter, the generated field is a HANA field.

1.3.2.2.1.1.1.2 Checks

Modeling users can define checks and register them in one or more functions later. This way, the system applies the checks to the results during an execution run.

Checks use selection conditions to detect specific records in the result data and append a message text and a message type to the application log.

The following fields are available:

- *Check*
  A unique identifier that initially has a predefined alphanumeric ID and can be used or changed by the modeler.
- *Category*
  Checks are classified as follows:
  - **Including Check**
    Validates whether the criteria in the *Selection* field appear in the result data. If the criteria is found, the system displays a message in the message log.
  - **Master Data Check**
    Validates whether the values of the fields are contained in the master data. For entries which are out of the master data range, a message appears in both the message log and in the column `FS_PER_MESSAGE_TEXT_` of the specific entry. This way, the user can see which entries are out of the master data range.
    Initial entries are also included in the validation. If there are no empty entries in the Master Data table, all initial values will trigger a message in the message log.
    If the Master Data table, in turn, does contain one or more empty entries, initial values for that function field are permitted.

    > **i** Note
    >
    > The `Selection` column is not taken into consideration for this type of check. All function fields will be checked against master data. If the user fills in fields in the *Selection* column, this will not be taken into consideration and a warning message will appear when saving the check.

- **Master Data or Initial Value Check**
  Validates whether the values of the function fields are contained in the master data. Initial entries are permitted and thus excluded from the validation. As a result, no message will ever be shown for initial entries.

  > **i Note**
  >
  > The `Selection` column is not taken into consideration for this type of check. All function fields will be checked against master data. If the user fills in fields in the *Selection* column, this will not be taken into consideration and a warning message will appear when saving the check.

- **Excluding Check**
  Validates whether the criteria in the *Selection* field appear at least once in the field values. If the criteria is not found at all, the system displays a message in the message log. This way, the user will notice if an important, must-have value is omitted from the value set.

- *Selection*
  Defines a selection. If the selection condition is satisfied, the system will display a message type.
- *Message Text*
  A message that is written to the application log when a check/condition is satisfied.
- *Message Type*
  A message type can be an error, abort, warning, information or status. If the check condition is satisfied, the system behaves as follows:
  - "Error" produces a result and a business event is created.
  - "Abort" produces a result and a business event is created.
  - "Warning" produces a result with a warning notification on the application log.
  - "Information" produces a result that is displayed in the application log.
  - "Status" behaves the same as "Information", but creates a different message type.

> **i Note**
>
> - Only one message from one check will be shown per entry even if the entry qualifies for multiple checks.
> - The priority among the different category checks is as follows:
>   1. Including Checks
>   2. Master Data Check OR Master Data or Initial Value Check
> - Excluding Check messages are displayed always, regardless of other check categories.
> - The priority among the same category checks is the order in which they appear on the *Environment – Checks* tab.

Related Information

Master Data Checks on Environment Level

1.3.2.2.1.1.1.3 Partitioning

You can use partitioning to enable and define the parallel package processing of data. Partitioning splits the dataset into subsets and triggers the calculation logic for several parallel threads at the same time. It can be used to handle datasets with more than 2 billion records or actively manage RAM and CPU usage.

For more details, see Parallelization and Partitioning [page 20].

1.3.2.2.1.1.1.4 File Formats

File format definitions are centrally defined in the environment and are referred to by *File Adapter* functions for data import and export.

Adding a File Format

To add a file format entry that can be used by the *File Adapter* function, follow the steps below:

1. On the *Environment Details* screen, choose the *File Format* tab.
2. Choose *Edit* in the upper right corner of the *Environment* section.
3. Choose *Add*.
4. The *Add Details* window appears, where you have to define the following:
   - *File Format*: A unique ID given to the file format entry, such as “CSV”, “TXT”, “CSV_SERVER0”, and so on.
   - *Description*: Text to describe the file format, for example “CSV from Server 0”.
   - *File Name*: This is an optional field where you can define a specific file name to be consumed by the *File Adapter* function. If the entered file name does not exist on the server, nothing is displayed on the *Server File* tab of the *File Adapter* function. If you want to choose from a list of files on the *Server File* tab, leave this field empty.
5. Choose *OK*.
6. Continue by defining the following:
   - *Skip First Row*: The number entered here is the starting point for the system to read the data. If you leave this empty, the system uses the first data record of the file being consumed as the first row.

     > **i Note**
     >
     > In the *File Adapter* function, the *Header Row* field allows you to specify the field names. If you leave this empty, the system names the fields “Column_1”, and so on.

   - *Record Delimited by*: Character that separates data record rows. As an example, you can choose to use next line (\n) if the next set of records is in the next row.

     > **Example**
     >
     > FIELD1;FIELD2;
     >
     > PROD01;CUST01;

PROD02;CUST02;

Another example is to use the equal sign (=) if sets of records are divided by the = character.

> **Example**
>
> FIELD1;FIELD2;=CUST;PROD;=PROD01;PROD02;

- *Field Delimited by*: Character that separates columns. Usually in a CSV file fields are separated by a comma (,) or semi-colon (;).

> **Example**
>
> FIELD1;FIELD2;
>
> PROD01;CUST01;
>
> PROD02;CUST02;

- Whitelist Path: Directory or path on your application server where the file is available.

> **Note**
>
> For more information about whitelist path setups, see SAP Note 2910484.

7. You also have the option of defining the following:
   - *Optionally Enclosed by*: When using this optional field, rows of data can be separated into columns.

> **Example**
>
> In the following examples the *Field Delimited By* field is set to ";" and the *Optionally Enclosed by* field is set to ".
>
> - **No Optionally Enclosed by**
>
> | Scenario | Input | Returned Column 1 | Returned Column 2 |
> |---|---|---|---|
> | Scenario 1 | PRO;D01 | PRO | D01 |
> | Scenario 2 | ""PRO";"D01"" | "PRO" | "D01" |
>
> - **With Optionally Enclosed by**
>
> | Scenario | Input | Returned Column 1 | Returned Column 2 |
> |---|---|---|---|
> | Scenario 1 | "PRO";"D01" | PRO | D01 |
> | Scenario 2 | "PRO;D01" | PRO;D01 | |
> | Scenario 3 | ""PRO";"D01"" | "PRO";"D01" | |

   - *Time Format*: Enter the time format to be used, for example "HH24:MI:SS".
   - *Timestamp Format*: Enter the timestamp format to be used, for example "YYYY-MM-DD HH24:MI:SS.FF7".
   - *Date Format*: Enter the date formats used in the file, for example "YYYY-MM-DD".

- *Thousand Separator*: Enter the thousand separator used in the amount fields, for example dot (.) or comma (,).
- *Decimal Separator*: Enter the decimal separator used in the amount fields, for example dot (.) or comma (,). This must be different from the specified thousand separator.

8. Choose *Save*.

Removing a File Format

Deletes a file format that is no longer relevant to the environment.

To remove the file format, follow these steps:

1. Select the row to be deleted and choose *Remove*.
2. A confirm deletion window appears. Here, you can enter a comment if required.
3. Choose *Confirm*.

1.3.2.2.1.1.1.5 Conversion Types

Conversion type definitions are needed by the conversion function for currency and unit conversions.

You need to define which references in the system are used to run the conversion function for currency and unit conversions.

> **Example**
>
> For example, category (unit or currency), tables (T006 or TCURR...), from which client, schema, conversion methods, rates, and market-relevant data.

Adding a Conversion Definition

To add a definition, follow the steps below:

1. On the *Environment Details* screen, choose *Conversion Types*.
2. Choose *Edit* in the upper right corner of the *Environment* section.
3. Choose *Add*.
4. Define the following:
   - *Conversion Types*: This is a unique ID that must be provided by the modeler (for example “CONV0001”).
   - *Description*: To distinguish between rules, enter a description. Once the description has been created, editing is still possible.
   - *Conversion Category*: Specifies the conversion category (unit and currency).
5. Choose “OK”.

6. Continue defining the fields. Since the conversion function uses SAP S/4HANA Conversion, for more information about the other fields available in the conversion definition, see the following:
   - CONVERT_CURRENCY Function
   - CONVERT_UNIT Function
7. Choose *Save*.

Removing a Conversion Definition

Deletes a conversion definition that is no longer relevant to the environment.

To remove an entry, follow these steps:

1. Select the row that you want to delete and choose the *Remove* button.
2. A confirm deletion window appears. Here, you can enter a comment if required.
3. 3. Choose *Confirm*.

1.3.2.2.1.1.1.6 Advanced

The advanced settings allow you to define standard integration scenarios using the following fields:

- **DB Connection Name**
  Here, you need to register the NetWeaver database connection to the underlying SAP HANA database. Since a user and password is always attached to a database connection, it indirectly specifies the authorizations and therefore which data and views are available. By default, this is the standard DBCON connection.
- **Call Back RFC Destination / RFC Destination**
  This allows you to connect an SAP ERP or SAP S/4HANA system to SAP Finance and Controlling. If you define this, the remote function adapter can post documents to an SAP S/4 HANA system or replicate master data from it (for example, to replicate an InfoObject from a remote system back to SAP Profitability and Performance Management).
  To maintain this in the Modeling Environment, select *Environment Button* > *Advanced Tab* > *RFC Destination* or choose *Call Back RFC Destination* and enter “RFC destination of PaPM client”.

1.3.2.2.1.2 Function Hierarchy

The modeling environment allows you to construct and define a model by adding and connecting multiple functions to a common network. The output of a function can be the input of other functions and thus contribute to the logic of the model.

You can arrange these functions in a function hierarchy, which is displayed on the left-hand side of the screen. You can place functions under *Calculation Unit*, *Description* and *Condition* functions only. This means, you cannot place a function under *Derivation*, *Join* or *View*, for example.

This hierarchy has no effect on the logic of the model and simply improves readability. You can add, remove, change, and copy functions. In change mode, the hierarchy function is locked against changes by other users, and changes are made persistent when you save. Other users can see these changes once they refresh the hierarchy function or switch to change mode themselves. Where-used lists and a network diagram can visualize the logical dependencies of the output/input relationships.

Procedure

1. In the modeling overview, select the modeling environment that you want to process and choose *Continue*.
2. The system directs you to the *Modeling Environment* window where you can process the following activities:
   - Add a function [page 64]
   - Remove a function [page 66]
   - See the function attributes [page 66]
   - Activate a function
     You can choose the activation button to activate a single or multiple functions at the same time after selection.The system then generates runtime artifacts for a function, such as stored procedures.
   - Copy a function [page 66]
   - View the where-used list for a function
     If you choose the *Where Used* button, the system shows you where the selected function is used as input.

1.3.2.2.1.2.1 Add Function

You can use the *Add Function* button to create a new function:

1. Choose an entry from the list of functions as the starting point of the description or function that you are creating.

   > **i** Note
   >
   > By default, the system selects the calculation unit as the starting point.

2. Choose *Add (+)* to open the *Add Function* window.
3. Make entries in all the fields as follows:
   - **Add Level**
     You can choose one of the following options:
     - "Same Level": This option means that the function that you create is structured or created at the same level as the selected entry.
     - "One Level Below": You can only use this option if you have selected a calculation unit or description function before choosing *Add (+)*.

     If you have selected a function before choosing *Add (+)*, you can only add a function on the same level and the system will not allow you to add a function one level below.
   - **General Tab**

1. **Function**
   This is an ID that is initially numeric, but that you can change to alphanumeric form if required. It is permanently assigned to the function. Once it has been created, it is not possible to edit the function ID.
2. **Description**
   Enter a description here to name the function. You can adjust this description later, if necessary, by choosing *Edit* in the *Modeling Environment* screen.
3. **Function Type**
   You can choose one of the available functions used for modeling. For more information see the Functions [page 107] section.
4. **Event Handling**
   You can choose one of the following options:
   - "Logging": Registers function processing errors in the application log after processing.
   - "Management": This allows all unprocessed records to be further processed using the *My Events* application.

     > **Example**
     >
     > If the allocation function has an unassigned item, set *Event Handling* to "Management".
5. **Processing Type**
   You can choose one of the following options:
   - "Sub Function": Functions that do not populate data records in autogenerated DDIC tables (Y tables) unless they are assigned as an activity under a process template, or a run process has been initiated. Choose this option if a function is just meant to be an interim function in the overall modeling logic.
   - "Executable": You assign this to functions that need to populate data records in the modeling logic.

     > **Example**
     >
     > The *Writer* function used as an input for another function, or functions that are to be used as inputs of a *Query* function.
6. **Partitioning**
   For more information, see Parallelization and Partitioning [page 20] .

- **Advanced Tab**
  1. **Package Selection Section**
     To optimize your parallelization setup while consuming data records from data sources such as the *Results Data* component, you can define a package selection by adding at least one field that is present in the data source.
  2. **Package Parameters Section**
     Maintain this section with parameters to transfer values from an external caller that will substitute those parameters in the configuration logic.

4. Choose *OK* for the changes to take effect.

1.3.2.2.1.2.2 Remove Function

Deletes a function that is no longer relevant to the environment.

To remove a function, follow the steps below:

1. Select the function to be deleted and choose the *Remove* button.

   > **i** Note
   >
   > If the function is used as an input function, a pop-up window appears indicating where it is used. The function must first be removed as an input before you delete it.

2. A *Confirm Deletion* window appears. Here, you can enter a comment if required.
3. Choose *Confirm*.

1.3.2.2.1.2.3 Function Attributes

The following function attributes are available:

- **General Tab**
  The content of this tab has the same information as that which is used when you create or add a function.
- **Advanced Tab**
  The content of this tab has the same information as that which is used when you create or add a function.
- **Runtime Attributes**
  This is a display only tab, which shows the generated runtime artifacts of a function, such asthe following:
  - Generation State
  - DDIC Table
  - Stored Procedure
  - Info Provider
  - Process Chain
  - Query
  - Planning Function
  - Time Stamp
- **Analysis Layouts**
  This contains the saved layout on the *Analyze* screen of a function that is included in a transport that can be moved to the next system.

1.3.2.2.1.2.4 Copy

When you copy a function, all the configuration settings made in this function are automatically copied to the new one. By default, the copied function has the same function ID and description as the original function.

> **i Note**
>
> Since the function ID needs to be unique in the environment, you can opt to change the function ID manually. Alternatively, you can press the *New IDs* button to generate a numeric value to be used as the function ID of the copied function.
>
> You can also change the description to meet your naming requirements for the new function.
>
> When you copy an environment, the newly created environment is not automatically activated.

Procedure

To copy a function, follow the steps below:

1. Select the function to be copied and choose *Copy*.
2. The system displays the *Copy Function(s)* window, where you have to enter the following information:
    1. **Target Environment**
       This is the environment in which the newly copied function is created.
    2. **Target Function**
       The target function is the basis on which the copied function is positioned.
    3. **Copy Level**
        - "Same Level": Creates a function structured on the same level as the target function.
        - "One Level Below": Creates a function structured below the calculation unit or a description function which we also call a node.
    4. **Copy Type**
        - "Modeling Copy Functions": Creates a new entry of the copied function in the target environment.
        - "Modeling Merge Functions": The configuration of the selected function is merged into an existing function in the target environment.
3. Confirm your changes to copy the function.

1.3.2.2.1.3 Function Details

When you select a function in the *Hierarchy* function, the system displays the function details on the right-hand side of the screen.

Depending on the function type, certain functions are available in display mode to run a function, to analyze or to show a result, for example.

In edit mode, the function is locked to prevent changes by other users, and your changes are persisted when you save. Other users can see these changes once they display the function details or switch to edit mode themselves.

1.3.3 Execution

1.3.3.1 Execution Overview

The execution overview displays various key performance indicators that are relevant for execution, such as biggest and smallest runtimes and processed data volumes.

Key Features

The following key features are available:

Key Feature	Use
Execution Key Performance Indicators	The execution overview provides users with an overview of the runtime, data volumes, and other useful information about the execution of models.
Key Performance Indicator Graphs	All key performance indicators are displayed in a graphical format. Each graphic is interactive and the user can navigate from the elements to the corresponding environment, the application monitor, the process monitor, and the modeling history. Graphics can also be displayed in a tabular format.

> **i Note**
>
> To use the full functionality of *Execution Overview*, we highly recommend to launch SAP Profitability and Performance Management via Fiori Launchpad.

Related Information

For more information about environments, see Financial and Business Modeling Entities [page 7].

For more information about how to define processes, activities, parameters, and selection fields, see Calculation Unit [page 337].

For more information about how to deploy processes, see Manage and Deploy Processes [page 93].

For more information about how to monitor processes, see Process Monitor [page 98].

1.3.3.2 My Activities

You can access the current activities of your team that need to be processed.

This application allows you to execute process activities, change the *Activity State* (complete, submit, approve, reject), change parameters and selections (for simulation run type only) and change comments. Various actions are available in the *GoTo* menu: You can choose *Application Log*, *Business Event Management* or *Modeling* for the selected activity.

The application does not display process instances and their activities that are assigned to other teams and that are not relevant for you. The system displays only deployed process instances.

> **i** Note
>
> You use the *My Activities* application for execution, and the *Manage and Deploy Processes* application for process instance management.

Key Features

The following key features are available:

Key Feature	Use
Processes	Processes are displayed in a hierarchy on the left together with the environment to which they belong. By default, only the current processes that need attention from the user are displayed. All processes can be displayed, including finished processes. A progress indicator shows how many of the activities are already finished.

Key Feature	Use
Activities	If a process is selected, the relevant activities for the user are displayed on the right. Activities can require two types of attention: 1. Input/output This type of activity requires manual user interaction because they either display data for review or allow data input. In both cases, users launch an analytic report to access the data. 2. Execution This type of activity triggers automatic logic and calculations. In this case, users run a function to produce interim or final results. By combining both types of activities, complex decentralized processes are structured that can involve multiple teams and various manual and automatic steps in parallel or in sequence, including an optional business workflow with the principle of dual control.
Parameters	All the parameters that are relevant for the execution of the process activities are listed here with their values. If the process type is "Simulation", the parameters can be changed at any time during the execution of activities. If not, they are fixed during the deployment of the process.
Selections	All the selections that are relevant for the execution of the process activities are listed here. If the process type is "Simulation", the selections can be changed at any time during the execution of activities. If not, they are fixed during the deployment of the process.

The following authorization checks are implemented:

- Display
  A user without this authorization can see all the activities in *My Activities*. However, if he chooses the button *Goto* > *Configuration*, he will not be able to see the function ID (FID) in *Modeling*.

  > **i Note**
  >
  > The authorization check is performed in *Modeling*.

- Run
  A user without this authorization is not allowed to run executable activities. Those activities are displayed in the list but cannot be run.

  > **i Note**
  >
  > - When both *Performer* and *Reviewer* fields are empty, the authorized user to execute the *Run* button should have at least one team assignment.

- When both *Performer* and *Reviewer* fields are filled, the authorized user to execute the *Run* button should belong either to the performer or reviewer group.
- When the *Performer* field is filled and the *Reviewer* field is empty, the authorized user to execute the *Run* button should belong to the performer group.

- Launch
  A user without this authorization is not allowed to perform the actions *Launch* and *Launch in Excel* for specific activities. Those activities are displayed in the list but cannot be launched in *Analyze* screen or Excel.

  > **i Note**
  >
  > - When both *Performer* and *Reviewer* fields are empty, the authorized user to execute the *Launch* and *Launch in Excel* button should have at least one team assignment.
  > - When both *Performer* and *Reviewer* fields are filled, the authorized user to execute the *Launch* and *Launch in Excel* button should belong either to the performer or reviewer group.
  > - When the *Performer* field is filled and the *Reviewer* field is empty, the authorized user to execute the *Launch* and *Launch in Excel* button should belong to the performer group.

- Complete
  A user without this authorization is not allowed to set the activities to “Complete”. Those activities are displayed in the list but cannot be completed.

  > **i Note**
  >
  > - When both *Performer* and *Reviewer* fields are empty, the authorized user to set the activities to “Complete” should have at least one team assignment.
  > - When both *Performer* and *Reviewer* fields are filled, the authorized user to set the activities to “Complete” should belong either to the performer or reviewer group.
  > - When the *Performer* field is filled and the *Reviewer* field is empty, the authorized user to set the activities to “Complete” should belong to the performer group.

Related Information

For more information about displaying and editing data, see Analytics Component [page 76].

For more information about how to define processes, activities, parameters, and selection fields, see Calculation Unit [page 337].

For more information about how to deploy processes, see Manage and Deploy Processes [page 93].

For more information about how to monitor processes, see Process Monitor [page 98].

1.3.3.3 My Events

You can access the current exceptional business events that occurred during the execution of activities and need to be processed.

The situation handling of exceptional business events can be done manually or automatically. In the latter case, you can define an automatic resolution rule which is then applied every time such a business event occurs, so that no manual interaction is required.

The application does not display other business events where the processes are assigned to other teams and are not relevant for the you.

Key Features

The following key features are available:

Key Feature	Use
Processes	Processes are displayed in a hierarchy on the left together with the environment to which they belong. By default, the system displays only the current processes that need your attention. All processes can be displayed, including finished processes. You can expand the hierarchy to drill down further to see the activity and the function of the activity in which the business event occurred. Additional indicators show if and how many automatic rules have already been defined, how many records are affected, and what volume (the sum of the key figures of these records). Both quantity and volume give a first indication of how material the business events are.

Key Feature	Use
Events	The business events are displayed on the right in a list with additional information about the state of the event, the message text, the affected quantity of records, and volume. You can select an event to view the detailed data and decide what steps to take to resolve the situation: 1. Event The event will not be handled and left in an open status. 2. Adjustment You can adapt and correct the underlying data for this event and run the corresponding activity again. This step can be repeated until the situation has been resolved and the quantity and volume shows 0. Technically, the business event handling does not change the data of a data source. Instead it applies a one-time rule on the input of a function to adjust the data accordingly. 3. Transmit The erroneous record will be moved directly to the result without adjustment. 4. Ignore If the event is not material or otherwise important, the event can be ignored. No partial restart of an activity is necessary in this case.
Rules	Automatic business event rules are managed in the same way as events. The only difference is that an adjustment rule is permanent and applied automatically each time in the future when a corresponding event occurs.

Related Information

For more information about the definition of business event fields, see Calculation Unit [page 337].

1.3.3.4 My Reports

You can access the current reports that are defined on top of processes and can also create new reports. The application does not display other business events where the processes are assigned to other teams and are not relevant for you.

Key Features

The following key features are available:

Key Feature	Use
Report Management	Reports are displayed on the left in a hierarchy with the environment to which they belong. Reports can be either private for a user or accessible for a team. The main purpose of reports is to provide dynamic reports and what-if simulations that can cover multiple processes and activities in an environment. You can select and launch a report, which will open the simulation and reporting application.
Elements of a Report	Reports consist of one or more elements, where each element refers to a process. Reports and elements inherit all their settings from the underlying process and activities like default layouts, teams, and authorizations. If processes are included and the process has the type "Simulation", the launched report can be used for a what-if simulation as all parameters are available for changes and activities with the type "Execution" can be triggered to run.

Procedure

In the client where SAP Profitability and Performance Management is installed, choose *SAP Menu* > *Profitability and Performance Management* > *Execution* > *Start My Reports* or launch the transaction code `/NXI/P1_Report`. The system opens the *My Reports* browser window where you can process the following activities in **Edit** mode.

Assign a User Group

You can assign a user group to a report. This report is then only visible to the members of that specific group or team. Follow the steps below to assign a user group:

1. Select a report from the report list to be the starting point for the user group assignment and choose *Edit*.
2. Choose *Change* . The *Change Report* window appears.
3. In the *User Group* field, select the user group that you want the report to be visible to.

> **i Note**
>
> If you do not assign a user group to a report, the report is visible to all users that are assigned to any group or team.

4. Choose *Save*.

Copy a Report with User Group

You can copy an existing report and assign a user group. The copied report is only visible to the members of that specific group or team. Follow the steps below to copy a report:

1. Choose the report you want to copy from the report list.
2. Choose *Copy*. The *Copy Report* window appears.
3. Fill out the following in the *Copy to* section:
   - *Environment*: Specify the ID for the environment that the report is copied to.

     > **i Note**
     >
     > The environment details are initially auto-filled with the current environment information. You can change it, if required.

   - *Report*: Specify the ID for the created report.
   - *Description*: Specify a description for the created report.
   - *Is Private*: If you select this checkbox, the report is available only to the report creator or to the user who selected the checkbox .
   - *User Group*: Specify the user group that you want to be able to see the report.

     > **i Note**
     >
     > If you do not assign a user group to the report, it is visible to all users that are assigned to any group or team.

4. Choose *Copy*.

Related Information

For more information, see Simulation and Reporting [page 76].

1.3.3.4.1 Simulation and Reporting

The application runs reports for execution users and gives them access to all the information for the report elements. By default, dynamic reporting capabilities are included to execute drill-downs and adapt the layouts of all the elements of the report. If simulation in the underlying processes is also enabled, what-if simulation is available in the report as well.

Key Features

The following table explains the key features available.

Key Feature	Use
Elements and Parameters	A list of all element titles and parameters available in the report is shown on the left-hand side of the screen. If what-if simulation is enabled, parameters can be changed and the execution of activities is also possible.
Charts and Tables	All input/output activities are visualized in chart or table format on the right-hand side of the screen. This visualization uses the standard analytics component application so that all of its features are available for each report element.

Related Information

For more information, see Analytics Component [page 76].

1.3.3.4.1.1 Analytics Component

The analytics component is the standard application to visualize data. It allows interactive self-service reporting, where you can display data in data grids and charts. If the underlying data model and the query function enable data editing, you can also modify and input data.

Key Features

The following table explains the key features available.

Key Feature	Use
Dimensions for Navigation	The list of dimensions for navigation can be shown or hidden. You can decide which dimensions appear on the row and column axes. The following additional options for manipulating each characteristic are available in the context menu: • Sorting • Filtering • Use of master data hierarchies • Display of IDs or texts for characteristic data
Data Grid	You can manipulate the data grid using the options in the context menu, various toolbar buttons and the collapse or expand icons of the hierarchy nodes. If the underlying query is input-enabled, you can also edit and save data.
Chart	The chart component provides a large selection of different and highly configurable graphs that provide visual representations of business data. The chart component also provides an out-of-the-box drill-down feature for interactive analysis.
Value Flow	The value flow diagram provides modern visualizations, especially to display the flow of values and money between dimensions. The value flow diagram also provides an interactive drill-down feature.

Related Information

For more information about the analytics component, see Analytics Component.

1.3.3.5 Studio

SAP Profitability and Performance Management comes with an optional, visual user interface, which can be accessed via the *Studio* tile in the SAP Fiori Launchpad.

From SAP Profitability and Performance Management Studio, you can access the following applications:

Application	Use
Environment [page 78]	Allows you to maintain default and custom connections. This is the home screen for accessing your calculation models.

Application	Use
Modeling [page 79]	Allows modeling users to design and test calculation models.
Process Management [page 84]	Gives you a central access point to your processes and activities.
Report Management [page 86]	Provides a central access point for your reports and what-if simulations.

There are also several screens that can be accessed from the SAP Profitability and Performance Management Studio, such as:

Screen	Use
Visualize Screen [page 88]	Displays highly aggregated data of a *Query* function in various chart types.
Comparison Screen [page 91]	Allows the comparison of two environment versions.

Procedure

Follow the steps below navigate the *Studio* application:

1. In the client where SAP Profitability and Performance Management is installed, enter transaction code `/N/UI2/FLP`.
2. The SAP Fiori Launchpad window appears.
3. Choose *Studio*.

Related Information

For more information about the enhancements and new features of the Visual Modeler, see 3127975.

1.3.3.5.1 Environments

1.3.3.5.1.1 Environment

The *Environment* application gives access to the list of available environments. It also allows you to create, edit or delete an environment or directly access environments.

Procedure

Follow the steps below to work with the *Environment* application:

1. In the client where SAP Profitability and Performance Management is installed, enter transaction code `/N/UI2/FLP`.
2. The SAP Fiori Launchpad window appears.
3. Choose *Studio*.
4. The system displays a new screen showing all the nodes and environments available. The next steps provide three main activities that can be done in the environment, such as *Add*, *Delete* and *Edit*.

Adding a Node or an Environment

1. Choose + (*Add*) from the toolbar, then choose between 🗀 (*Node*) or 🖧 (*Environment*).

   > **i Note**
   >
   > An environment is not allowed to have sub-environments. If you want to structure your environment list, use a node. Nodes can hold environments and sub-nodes.

2. Provide a description, environment ID and version in the corresponding fields.

   > **i Note**
   >
   > *Version* is only needed for environments.

3. Choose ✓ (*Confirm*).
4. Choose 🖫 (*Save*).

Deleting a Node or an Environment

1. Select the environment or node to be deleted.
2. Choose 🗑 (*Delete*) from the toolbar.
3. Choose 🖫 (*Save*).

Editing the Description of a Node or Environment

1. Select the node or environment to be edited.
2. On the right-hand side of the environment or node, choose ✎ (*Edit*).
3. Change the description of the node or environment, then choose 🖫 (*Save*).

1.3.3.5.1.2 Modeling

The *Modeling* application enables you to model your own custom calculation logic with the help of modeling functions and entities that can be found in the modeling palette.

Below are the most usual sections and terminologies used in modeling application during configuration:

Modeling Section	Icon	Description
Palette		Offers all the modeling functions and tools you need to build your calculation logic, arranged by category.
Environment Details		Manages general entities such as fields, parameters, checks, conversion type and partitioning.
Calculation Unit		Upon creation of an environment, the system automatically creates a main calculation unit. A calculation unit is a container function that defines the functions relevant for execution using process templates and activities. It also specifies the parameters and selection fields that are required for execution. From a modeling perspective, it is a collection of objects, such as fields and functions.
Properties Panel		Used to further configure a selected function or entity.
General		Save, undo, redo changes, refresh the page
Edit		Cut, copy, paste or delete a function or modeling entity

Modeling Section	Icon	Description
Tools		The following buttons are available: • *Activate*: Activates a selected business function. • *Run*: Executes a selected business function. • *Show*: Shows the results of a selected business function in a tabular form. • *Analyze*: Analyzes the results of a selected business function in a pivot table or pivot chart. • *Analyze in Excel*: If the optional SAP Analysis for Microsoft Office Add-on is installed, you can analyze the results of a selected business function in Microsoft Excel in a pivot table or pivot chart. • *Visualize*: Presents the results of a selected business function in various chart and diagram types. For more information, see Visualize Screen [page 88]. • *Configuration*: Allows the navigation to the Modeling Environment. • *Messages*: Allows the navigation to the Application Monitor, where you can examine the messages across all environment versions. • *Historize*: Takes the current saved status of the whole environment configuration (including all field and function details) and saves this in the modeling history. • *Erase Data in All Buffers*: Allows the deletion of data within the selected model function.

Procedure

Follow the steps below to navigate to the *Modeling* application:

1. In the client where SAP Profitability and Performance Management is installed, enter transaction code `/N/UI2/FLP`.
2. The SAP Fiori Launchpad window appears.

3. Choose *Studio*.
4. Select an environment from the environment list.
5. In the *Go to* menu in the top right-hand corner of the screen, choose *Modeling*.

1.3.3.5.1.2.1 General Entities

The following general entities are available in SAP Profitability and Performance Management:

Entity	Icon	Description
Fields		A field is a unique technical name that initially has a predefined characteristic value which you can use as it is or change to a different value. When you create a field, you can choose from the following field types: • *Characteristic*: Used to identify key figures, and contains texts, codes, dates or numerical characteristic values. • *Key Figure*: Used for calculations and can contain natural numbers, integers, decimals or floating points. • *Unit*: Required to give meaning to the values of a key figure.
Parameters		Parameters are used to steer processes and calculations. They are used in formulas and calls of certain functions to influence the logic and operations applied.

Entity	Icon	Description
Checks		You can define custom checks here and register them in one or more functions later so that the system applies them to the results during an execution run. Custom checks use selection conditions to detect specific records in the results data, and append a message text and a message type to the application log. The following message type fields are available in the properties panel: • *Information* produces a result that is displayed in the application log. • *Status* behaves the same as the *Information* message type, but creates a different message type. • *Warning* produces a result with a warning notification in the application log. • *Error* produces a result and creates a business event. • *Abort* produces a result and creates a business event.
Conversion Types		Conversion type definitions are needed by the *Conversion* function for currency and unit conversions. You need to define which references in the system are used to run the conversion function for currency and unit conversions. **Example** Category (unit or currency), tables (T006 or TCURR...), from which client, schema, conversion methods, rates, and market-relevant data.

Entity	Icon	Description
Partitioning		You can use partitioning to enable and define the parallel package processing of data. Partitioning splits the dataset into subsets and triggers the calculation logic for several parallel threads at the same time. It can be used to handle datasets with more than 2 billion records or actively manage RAM and CPU usage. For more information, see Parallelization and Partitioning [page 20].

General Procedure

The following general procedure describes how you can add and edit a function in SAP Profitability and Performance Management:

1. On the left-hand side of the screen, choose (*Palette*).
2. The system displays a list of all available functions. Each function is shown with its name and its individual icon.
3. Some functions have multiple function types. In these cases, you can expand the function node in the list to see all the available options.
4. Select the function or function type you want to process so that it is highlighted with a blue frame.
5. Drag and drop the chosen function or function type into the modeling area in the middle of the screen (inside the main *Calculation Unit* function).
6. Once you have dropped the function or function type, choose (*Properties Panel*) on the right-hand side of the screen.
7. Configure the function.
8. When you have completed your configuration, choose (*Save*).
9. Choose (*Activate*).

1.3.3.5.1.3 Process Management

The *Process Management* application shows a visual representation of the process instances in the form of a Gantt diagram. You can inspect all settings of the process instances with their process parameters and selections here. You can also see the activities with their activity state, their assigned start and end date as well as their performer and reviewer team.

Key Features

The following key features available in the *Tools* and *Go to* menu:

Key Feature	Use
Run	If the process instance is in the deployed process state, you can execute a selected business function, so that the results are available for display, analysis and visualization afterwards.
Show	If the process instance is in the deployed process state, you can show the results of a selected business function in a tabular form, where the data can be filtered, sorted and also technical information per record can be displayed.
Analyze	If the process instance is in the deployed process state, you can analyze the results of a selected business function in a pivot table or pivot chart. See also *Analytic Component*.
Visualize	If the process instance is in the deployed process state, you can visualize the results of a selected business function in various chart and diagram types. See also Visualize Screen [page 88].
Configuration	Allows the navigation to the Modeling Environment.
Zoom	Allows you to zoom in, zoom out, zoom to fit, zoom to fit a week, month, and year perspective on the timeline.
Actions	If the process instance is in the deployed process state, the user teams (performer and reviewer) can perform the following actions: • Approve • Reject • Initialize Process
Modeling	Allows navigation to the *Modeling* application, where process environment versions are visualized in the form of a directed graph.
Reports	If you select a process template, you can navigate to the *Report Management* application, where the qualitative reporting and simulation is available.

Procedure

Follow the steps below to navigate to the *Process Management* application:

1. In the client where SAP Profitability and Performance Management is installed, enter transaction code `/N/UI2/FLP`.
2. The SAP Fiori Launchpad window appears.
3. Choose *Studio*.
4. Select an environment from the environment list.

5. In the *Go to* menu in the top right-hand corner of the screen, choose *Process*.

1.3.3.5.1.4 Report Management

The *Report Management* application allows the creation and consumption of reports as well as the execution of simulations if the underlying process template or process instance is set to process type "Simulation".

Static qualitative content like texts, images and videos can be combined with interactive visualizations, so-called report elements, based on input/output activities into a report. You can arrange multiple reports using separate tabs.

Dependent on the underlying process settings, the following edit restrictions apply:

1. **Reports on process templates**
   Reports are created and prepared based on process templates by the modeling user. In this case no edit restrictions in the report apply. When a process instance is created based on a process template, these reports are made available to execution users automatically.
2. **Reports on process instances of type "Simulation"**
   No edit restrictions to the execution user apply.
3. **Reports on process instances of type "Run"**
   The editing of the report is not restricted, as long as the process state is "Open". As soon as the process state is set to "Deployed", editing is no longer allowed . You can only change story filters and drill down or drill through in report elements.

Key Features

Dependent on the edit restriction, the following key features are available in the *General*, *Edit*, *Tools*, *Editor* and *Go to* menu:

Key Feature	Use
Palette	Shows the execution and input/output activities of the current process. From here you can drag the input/output activities as interactive report elements into the report to visualize them. You can also select the execution activities from here and trigger the execution or reexecution of an activity.
Save	Saves the report. The layouts of any embedded report element are not affected by this because they have to be saved separately.
Save All	Saves the report together with the report elements and also the layout of the embedded report element.

Key Feature	Use
Undo	Performs an undo of the last editing operation in the report. The undo/redo stack is cleared on *Save*.
Redo	Performs a redo of the last undone editing operation in the report. The undo/redo stack is cleared on *Save*.
Refresh	Allows to reload the *Report Management* application.
Cut	Cuts out the selected content of a report so that it is available for pasting.
Copy	Copies the selected content of a report so that it is available for pasting.
Paste	Pastes previously cut or copied content into the report at the current cursor position.
Simulate	Triggers the client-side simulation of a report, based on the maintained simulation script. After the simulation script is finished, the system updates the report elements accordingly.
Run	Triggers the server-side execution or reexecution of an activity if an execution activity is selected in the *Palette*, or if only one execution activity is available in the process.
Configuration	Allows the navigation to the Modeling Environment.
Messages	Allows navigation to the *Application Monitor*, where you can examine the messages across all environment versions.
Export to PDF	Allows saving of the *Report* tab into a PDF format.
Editor	All report editing features for the static content reside in the *Editor* menu. For example, paragraph style, font family, font size, formatting, text alignment, font color, font background color, numbered list, bulleted list, todo list, block quote, decrease indent, and increase indent. You can also insert images, tables and media, such as YouTube videos.
Modeling	Allows navigation to the *Modeling* application, where process environment versions are visualized in the form of a directed graph.
Process	Allows navigation to the *Process Management* application, where process instances are visualized in the form of a Gantt diagram.
Properties	The *Properties Panel* on the right-hand side of the screen shows all process parameters and process selections of the underlying process template or process instance. Story filters comprise all fields, which are in use by the embedded report elements and allow you to filter all report elements at the same time. The client-side simulation can contain a script, which can be executed via *Tools* > *Simulate*.

Procedure

Follow the steps below to navigate to the *Report Management* application:

1. In the client where SAP Profitability and Performance Management is installed, enter transaction code `/N/UI2/FLP`.
2. The SAP Fiori Launchpad window appears.
3. Choose *Studio*.
4. Select an environment from the environment list. Choose its corresponding number in the *Reports* column.
5. The *Navigate to Reports* section appears.
6. Choose the > (*Expand*) icon. The system displays the available reports based on a process template.
7. Choose the arrow icon or select a report to access *Report Management*.

1.3.3.5.2 Other Screens

1.3.3.5.2.1 Visualize Screen

You can trigger the *Visualize* application as a stand-alone application from *Modeling* as well as from *Process Management*. In both of these cases the system opens the application in a separate browser tab. The application can also be used for embedded report elements in *Report Management*.

Key Features

The basic set of features is identical in both cases:

Key Feature	Use
Annotations Palette	If you have set the *Enable Annotations* checkbox for the chart in the properties panel, the *Annotations Palette* is available on the left-hand side of the screen where you can comment and annotate the data in the chart.
Save	Saves the visualization. If the visualization is an embedded report element, the report itself is not affected by this, because it has to be saved separately.
Undo	Performs an undo of the last editing operation in the visualization. The undo/redo stack is cleared when you save.
Redo	Performs a redo of the last undone editing operation in the visualization. The undo/redo stack is cleared when you save.
Cut	Cuts out the selected content of a visualization so that it is available for pasting.

Key Feature	Use
Copy	Copies the selected content of a visualization so that it is available for pasting.
Layout	You can manage layout variants here. One layout is always marked as the default and the others can be selected as required.

Key Feature	Use
Properties Panel	The properties on the right-hand side of the screen contain a chart structure panel to customize the visualization of the following chart types: • Bar/column • Stacked bar/column • Area • Line • Time series • Heat map • Numeric point • Dimension value • Donut • Pie • Sunburst • Sankey • Sheet • Process • Relationship • Waterfall In addition to the data from the underlying input/output activity, additional calculated measures and calculated dimensions can be defined and filled using scripted calculations. These calculated measures and dimensions are then available and included in all further settings. The data can be sorted by multiple fields, and in ascending or descending order if needed. Under *Chart Settings* you can choose the measures and dimensions relevant for visualization. In addition you can define drilldown fields, which provide an easy and interactive way to go into details. For certain chart types you can enable annotations. This allows the individual commenting of data points and the drawing of lines. For time series, there is also a list of predefined financial and mathematical annotations available (for xample, trends and regressions). You can switch some chart types to "editable" which allows graphical changes of data. If data on aggregated level is changed (like a bar for example), the delta change is automatically distributed to all underlying data records. These data changes cannot be saved permanently, but are available for client-side simulations.

Key Feature	Use
	Under *Field Settings* you can maintain a value selection for each dimension . Based on the chart type, you can also define a pattern, color and icon. For measures, you can also maintain aggregations. Under *Styling* you can set the background color and a background image, title position, axis labels and legend positions.

Advanced Features

The following advanced features are only available for embedded report elements and specific chart types.

Drill-Through

Drill-through is available for all report elements of a report and can beaccessed by right clicking a data point or node in the chart (on tables long touch). It offers all other report elements and their layout variants as a target. All filters, story filters, visualization field setting filters as well as the dimension values of the selected data point or node are handed over to the target chart.

Flow Animation

The flow settings for *Flow Animation* are available for process and relationship diagrams. If a *Flow* field is selected in the properties, its values are offered in the diagram for flow animation. It is especially helpful to animate the flow through activities in Process Mining Analysis.

Procedure

Follow the steps below to trigger *Flow Animation*:

1. Open the context menu for a *Process* or *Relationship* diagram (right mouse click).
2. Choose *Flow Animation* and select one or more flow animation values.

1.3.3.5.2.2 Comparison Screen

Use the *Comparison* screen to compare two environment versions. Select two environment versions and choose *Compare* from the environment list to access the application.

Key Features

The following key features are available on the *Comparison* screen:

Key Feature	Use
Search	Locates content based on specific text entered in the search bar.
View	Shows the result of the comparison in unified or side-by-side view. It also includes the *Reset Code Blocks* feature which reverts to the default layout before expanding lines.
Export	Allows the comparison screen to be exported in PDF format.

Procedure

Follow the steps below to compare two environment versions:

1. In the client where SAP Profitability and Performance Management is installed, enter transaction code `/N/UI2/FLP`.
2. The SAP Fiori Launchpad window appears.
3. Choose *Studio*.
4. Select an environment from the environment list and choose its corresponding number in the *History Versions* column.
5. The *History Versions* section appears on the right-hand side of the screen, showing the history version details of the environment selected.
6. Select the history version of the environment by selecting the checkbox beside the date and time.
7. Choose the *Compare* button in the resulting pop-up dialog.
8. The system redirects you to the *Comparison* screen.

1.3.3.5.2.3 Data Maintenance Screen

Use the *Data Maintenance Screen* either by writing directly into the fields (using copy and paste) or by importing an to load data in the Excel file. function. You can insert the data in two ways: either by writing directly into the fields (using copy and paste) or by importing an to load data in the Excel file.

Key Features

The following key features are available:

Key Feature	Use
Upload File (Transport = “No”)	Uploads data using CSV: delimited and Excel files.
Undo	Reverses an earlier action.
Redo	Restores any actions that were previously undone.
Delete	Deletes the data in the selected cell.
Insert Row	Inserts a blank row below the selected row.
Minimize	Collapses the data editor window to a smaller view.
Maximize	Enlarges the data editor window so that it is displayed in full-screen mode.

Procedure

Loading Data via the Data Editor in the Function Properties

1. In the client where SAP Profitability and Performance Management is installed, enter transaction code `/N/UI2/FLP`.
2. The SAP Fiori Launchpad window appears.
3. Choose *Studio*.
4. Select an environment from the environment list.
5. In the *Go to* menu in the top right-hand corner of the screen, choose *Modeling*.
6. On the *Modeling* screen, select a *Model Table* function and expand the properties panel (represented by the icon) .
7. In the *Function Properties: Data Editor* section, you can add data records manually or upload them (if the *Transport Data* option is not active).
8. Choose (*Save*) .

1.3.3.6 Manage and Deploy Processes

This application allows you to manage process instances.

Process instances are based on process templates from the modeling application. Process instance management comprises the creation and deletion of process instances as well as the changing of process states.

> **i Note**
>
> Upon creating a process template, the system displays a warning message in the following cases:
>
> - If the *Performer* field of the activity is filled and the *Reviewer* field is empty, the system warns you that the dual control mechanism is not active.
> - If the *Performer* field of the activity is empty and the *Reviewer* field is filled, the system warns you that this combination is not allowed.

The default state after creation is "Open". Only processes in the state "Deployed" are visible in the *My Activities* application.

> **i Note**
>
> When creating a new process instance, the process ID is automatically generated if the number range interval is set for object `/NXI/1PROC`. If it is not set, the automatic process ID generator is not active and the *Process ID* field remains empty.

The *Manage and Deploy Processes* application allows you to change the following attributes:

- *Activity Description*
- *Start Date*
- *Due Date*
- Performer/Reviewer groups
- *Comments*.

If the *Process State* is "Open" or "Suspended", yu can also change parameters and selections. However, only. One of the most important features is the change of the activity state, especially the *Reset State* button changes the state of the selected activity to the initial value ("Open").

You cannot execute activities directly from the *Manage and Deploy Processes* application, but you can manage the settings listed above. You can execute activities from the *My Activities* application by choosing the *My Activities* button in the header of the *Manage and Deploy Processes* application.

The application manager runs processes and assigns activities to teams.

The following table describes the available key features.

Key Feature	Use
Processes	On the left-hand side of the screen, the system displays processes in a hierarchy together with the environment to which they belong. Processes can have various states: • "Open" Open processes can be changed and settings, for example, start dates, due dates, performer and reviewer team, parameters and selections can be maintained. Processes in state "Open" can be deployed so that the execution teams can start working on the processes. • "Deployed" Deployed processes are visible to the execution teams who can work on the activities in the *My Activities* application. Deployment processes need to be error-free to be completed. Otherwise, the system will suspend the process. Once a process instance is deployed, SAP Profitability and Performance Management sends an email notification to the performer team to inform about the newly deployed process. • "Suspended" Suspended processes are not visible to the execution team. In the same way as for open processes, changes can be applied to the settings, for example, due dates, parameters or selections. Then, the state can be set to "Deployed", "Aborted" or "Completed". • "Completed" If the activities of a deployed process have been run successfully, the process can be set to "Completed". • "Aborted" If a process is be terminated without success, the system sets the status to "Aborted". .

Key Feature	Use
Activities	If a process is selected, the activities are displayed on the right-hand side of the screen. Only if the process state is “Open” or “Suspended”, changes can be applied to the activity state, start date, due date, reviewer and performer team as well as to the parameters and selections. The activity can have various states: • “Open”: The activity is open for execution. • “Pending”: The activity is not open for execution yet because preceding activities are not finished yet. • “In Approval”: A dual control principle workflow is attached to the activity and this is not finished yet. • “Completed”: The activity is completed.
Parameters	All parameters that are relevant for the execution of the process activities are listed here with their values. Parameters can be changed only if the process state is “Open” or “Suspended”.
Selections	All selections that are relevant for the execution of the process activities are listed here with their values. The selections can be changed only if the process state is “Open” or “Suspended”.

Related Information

For more information about the application manager, see the Administration Guide for SAP Profitability and Performance Management.

For more information about the definition of processes and activities, see Calculation Unit [page 337].

1.3.4 System Reports

1.3.4.1 Application Monitor

The application enables you to inspect the messages that have been logged during activations and runs for every function within an environment. This helps you to find out if warnings or errors occurred and when.

The search, filtering and sorting of messages is supported. You can also export the application log to an Excel spreadsheet.

Key Features

The following table explains the key features available.

Key Feature	Use
Run Log	The application creates a unique log entry each time an individual function is generated and run. The log contains the following information: • A unique run ID • A status • A run set ID • An input set ID • A timestamp • The name of the user who executed the run or generated the function
Message Log	This contains the list of messages that are associated with every execution. The list of messages usually contains the following information: • The status and results of a run • Function-specific messages that are associated with a run (for example, unassigned items for allocation, records that were not transferred for the transfer structure/derivation) • The results of a generation

Related Information

For more information about the definition of custom specific checks that are logged in the application monitor, see Checks [page 58].

1.3.4.2 Process Monitor

The application enables you to examine all currently active and past processes. Search, filter and sorting of processes and activities is supported as well.

Key Features

The following table explains the key features available.

Key Feature	Use
Processes	Processes are displayed in a hierarchy together with A Progress indicator shows, how many of the included activities are finished.
Activities	If a process is selected, then on the right side the activities of the process are displayed.
Parameters	All parameters, which are relevant for the execution of the process activities are listed here together with their values.
Selections	All selections, which are relevant for the execution of the process activities are listed here together.

Related Information

For more information on how to define processes, activities, parameters and selection fields, see Calculation Unit [page 337].

For more information on how to deploy processes, see Processes [page 93].

1.3.4.3 Modeling History

This application enables you to trace and inspect the configuration changes to a model within an environment. This helps you to trace and audit who did what and when. Depending on the user authorizations, even historic versions of environments and functions can be restored.

The *Modeling History* application captures a snapshot of the configuration (as an XML file) only on important operations like “CREATE”, “DELETE” and “MANUAL”.

Key Features

The following table explains the key features available.

Key Feature	Use
Environment List	All current and historic environments are listed on the initial page. If an environment is deleted, it is no longer visible in the list.
History Versions	Once an environment is selected, a detailed list of all changes to the environment, functions and fields are displayed on the right-hand side of the screen. You can select and restore old versions of an environment, function or field.
Environment	If you select an environment, the system lists here all the operations executed within the environment together with an entry description, the history type, time stamp, user and name.
Function	All the operations executed on the function within the environment are listed here together with the timestamp, user and name.
Fields	All the operations executed for fields on the environment are listed here together with the timestamp, user and name.
Retrieve Version	Retrieves the selected history version entry.
Show Configuration	Displays the selected history version configuration in XML format.

The system creates the entries in the modeling history when the following prerequisites are met:

- An environment, function and field are either created, changed, deleted or transported.
- The *Historize* button is manually executed in the environment.

Procedure

In the client where SAP Profitability and Performance Management is installed, choose *SAP Menu* > *Profitability and Performance Management* > *System Reports* > *Show Modeling History* or launch the transaction code `/NXI/P1_MODEL_HIST`. The system opens the *Modeling History* window where you can process the following activities in edit mode.

Retrieve Function

You can use the *Retrieve* button to recover the specific environment, function or field:

1. Choose an environment from the environment list to be the starting point of the retrieval.
2. On the *History Versions* screen, choose one of the following tabs:
    - *Environment*: To retrieve a specific version of an environment.

- *Function*:To retrieve a specific version of a function.
- *Field*:To retrieve a specific version of a field.

3. The following columns appear on the *Function Details* section of the *History Versions* screen:
   - *Description*: This is the same as the environment name
   - *History Type*: Describes the type in which the historization was generated:
     - "CREATE": Is created when new or copied environments, functions or fields are saved.
     - "MANUAL": Historization can be triggered manually by choosing the *Historize* button in the modeling environment.
     - "CHANGE": This only logs the operation but does not capture a configuration snapshot. No information can be retrieved for this history type.
     - "DELETE": Is initiated when an environment, functions or fields are removed.
     - "TRANSPORT": Is created when an environment is transported to another system. Once it is initiated from the source system, the *Transport* history type appears.
     - "IMPORT": Is created when an environment has been imported from one system to another.
     - "RETRIEVED": Is created when an environment, functions or fields are recovered.
   - *Time Stamp*: Displays the system time when the *Historize* button was chosen and the historization was created
   - *User*: Is the ID of the user who performed the historization
   - *Name*: Is the full name of the user to whom the ID is assigned
4. Select the specific entry or version you want to retrieve. You can check the time stamp to find it.
5. Choose *Retrieve*.
6. The *Recover Environments* window appears.
   Fill out the following in the *Retrieve as Version* section when you retrieve an environment:
   - *Environment*: Define an environment ID for the retrieved environment .
   - *Version*: Specify the version ID that you want the retrieved environment to have.
   - *Description*: Specify a name for the retrieved environment.
   - *Parent Node*: Define under which node or description node the retrieved environment is placed.

   In addition, define the following when you retrieve a function:
   - *Parent Calculation*: Specify under which calculation unit the function is placed.
   - *Function*: Specify the function ID for the retrieved function.
7. Choose *OK*.
   To validate the retrieved item, navigate to the details you set in the *Retrieve as Version* section.

Manual Historization

To perform manual historization, select the entry to be historized in the function hierarchy on the *Modeling Environment* tab and choose *Historize*. The system then displays the following message: "Historization successfully completed".

Related Information

For more information about modeling, see Modeling Environment [page 48].

1.3.5 Tools

1.3.5.1 Activate Function

You can use this tool to activate a function without having to access the modeling environment.

The following fields are available:

Fields	Description
Environment	This is a 3-digit alphanumeric ID that is permanently assigned to the environment. Use the F4 help to display all environments configured in the system and client. This field is used as a filter to specify an environment
Version	This is a 4-digit alphanumeric ID that is permanently assigned to the environment. You can use this field as a filter to reduce the number of records so that the system checks only within the specified version of the environment.
Function	The identifier of the function where the data to be activated is located. Use the F4 help to display all the functions configured in the specified environment and version.
Run Type	Allows you to select which run type to perform. You can choose the following options: • Activation (ACT) Generates the procedure of the environment/function in real time • Activation Simulation (ACT_SIMU) Runs the tool in test mode. The system issues configuration errors upfront without generating its procedure.
Show Model Flow Diagram	Allows you to view the Model Flow Diagram of the environment.
Active necessary func. only	Activates all executable function, data sources and process templates. This can only be used when you activate an environment.

Procedure

In the client where SAP Profitability and Performance Managementis installed, choose *SAP Menu* > *Profitability and Performance Management* > *Tools* > *Activate Function* or launch transaction code `/NXI/P1_FW_ACTIVATE`.

1. The *Performance Management Activate* window appears.
2. Make entries in or choose the following required (*) and optional fields:
   - *Environment
   - *Version

- Function
- Run Type

3. Choose one of the following options:
   - Show Model Flow Diagram
   - Activate necessary func. only

After you have made the relevant selections, the program can be executed immediately, either in the background or as a scheduled task. Continue with the following steps.

Execute Immediate (F8)

1. Choose *Execute* or F8.
2. The *Display Logs* window opens.

Execute in Background – Immediate

1. Choose *Program* > *Execute in Background* on the main menu.
2. On the *Background Print Parameters* screen, choose *Continue*.
3. On the *Start Time* screen, choose *Immediate* and *Save* consecutively to run the program in the background.
4. To see the progress of the background run, launch transaction code **SM37**.

Execute in Background – Scheduled

1. Choose *Program* > *Execute in Background* on the main menu.
2. On the *Background Print Parameters* screen, choose *Continue*.
3. On the *Start Time* screen, choose *Date/Time*.
4. The *Date/Time* section appears, and you can specify the date and time that you want to schedule for the run.
5. To see the progress of the background run, launch transaction code **SM37**.

1.3.5.2 Run Function

This tool helps you to run functions without having to access the modeling environment.

The following fields are available:

Field	Description
Environment	This is a 3-digit alphanumeric ID that is permanently assigned to the environment. Use the F4 help to display all the environments configured in the system and client. This field is used as a filter to specify an environment.
Version	This is a 4-digit alphanumeric ID that is permanently assigned to the environment. You can use this field as a filter to reduce the number of records that the system checks to within a specified version of the environment.

Field	Description
Calculation Unit	This is the ID of the calculation unit where the function relevant for execution is located. Choose F4 to display all the calculation units configured in the specified environment.
Process	Calculation-unit-wide unique ID of a process template. This can be referred to in process management to instantiate processes.
Activity	Process-template-wide unique ID of an activity. This can be referred to in report elements.
Function	The ID of the function relevant for execution. Choose F4 to display all functions configured in the specified environment and version entered above.
Run ID	The ID of the run relevant for execution.
Business Event	The ID of the business event relevant for execution.
Run Type	Allows you to select which run type to perform. You can choose from the following options: • Run (RUN) Executes the run of the chosen function in real time • Run Simulation (RUN_SIMU) Executes the run of function in test mode
Package	The ID of the package of a function relevant for execution.
Package Parameter	Specifies a finite value for the parameter to be incorporated in function execution.
Package Selection	Specifies a field filter (selection) to be incorporated in function execution.
Run Mode	Specifies the mode that a function is run in. See Parallelization and Partitioning [page 20].

Field	Description
Synchronous Execution	Defines whether the run is carried out in synchronous or asynchronous mode: • When set to “Yes” (checked): If synchronous execution of the same function is triggered by two users simultaneously, only one of both users will be logged into the application monitor. The other user will not be able to work on the user interface until the execution of the run is completed. However, if the two users run different functions, both execution threads are logged in the application monitor. • When set to “No” (unchecked): The asynchronous execution mode is activated. Once the caller starts the execution, the run is executed in a separate process. The caller does not have to wait until the run is completed before starting another activity. In modeling, the run is always asynchronous. Once it gets started, you can continue modeling in other functions. **i Note** Before executing the asynchronous execution run, a dialog box is displayed where you need to choose a priority: • “Job Priority A”: High priority • “Job Priority B”: Medium priority • “Standard Priority C”: The system decides about the execution sequence based on the memory usage of the function. You will see the run details on the application log.
Show Result	Indicates whether the result of the run is displayed: • When set to “Yes” (checked): The system displays the results of the run in the SAP NetWeaver system. • When set to “No” (unchecked): Results do not appear in SAP NetWeaver system after execution

Procedure

In the client where SAP Profitability and Performance Management is installed, choose *SAP Menu* > *Profitability and Performance Management* > *Tools* > *Run Function* or launch transaction code `/NXI/P1_FW_RUN`.

1. The *Performance Management Run* window appears.
2. Make entries in or selections for the following required (*) and optional fields:
    - *Environment
    - *Version
    - Calculation Unit
    - Process
    - Activity
    - Function
    - Run ID
    - Business Event
    - Run Type
    - Package
    - Package Parameter
    - Package Selection
    - Run Mode
3. Choose one of the following options:
    - Synchronous Execution
    - Show Result

After you have made the relevant selections, the program can be executed immediately, either in the background or as a scheduled task. Continue with the following steps.

Execute Immediate (F8)

1. Choose *Execute* or F8.
2. The *Display Logs* window appears.

Execute in Background – Immediate

1. Choose *More* > *Program* > *Execute in Background*.
2. On the *Background Print Parameters* screen, choose *Continue*.
3. On the *Start Time* screen, choose *Immediate* and *Save* consecutively to run the program in the background.
4. To see the progress of the background run, launch transaction code SM37.

Execute in Background – Scheduled

1. Choose *Program* > *Execute in Background* on the main menu.
2. On the *Background Print Parameters* screen, choose *Continue*.
3. On the *Start Time* screen, choose *Date/Time*.
4. The *Date/Time* section appears, where you can specify the date and time that you want to schedule for the run.
5. To see the progress of the background run, launch transaction code SM37.

1.3.5.3 Delete Temporary Data

This tool deletes Y-tables, including the data produced by functions inSAP Profitability and Performance Management.

The following fields are available:

Fields	Description
Environment	This is a 3-digit alphanumeric ID that is permanently assigned to the environment. Use the F4 help to display all the environments configuredin the system and client. This field is used as a filter to specify an environment.
Version	This is a 4-digit alphanumeric ID that is permanently assigned to the environment. You can use this field as a filter to reduce the number of records that the sytem checks to within the specified version of the environment.
Calculation Unit	The ID of the calculation unit where the data to be deleted is located.
Function	The ID of the function where the data to be deleted is located. Choose F4 to display all the functions configured in the environment and version specified above.
Run In Simulation	If you choose this option, the system executes the program without actually deleting the data. Otherwise, the system executes the deletion.

Procedure

In the client where SAP Profitability and Performance Management is installed, choose *SAP Menu* > *Profitability and Performance Management* > *Tools* > *Delete Temporary Data* or launch transaction code `/NXI/P1_FW_DEL_TDATA`.

1. The *Delete Temporary Data* window appears.
2. In the *Select Function* section, make entries in the following optional fields:
3. In the *Simulation* section, you can choose whether the program is run in actual mode or in simulation mode.

Once you have made the relevant selections, the program can be executed immediately, in the background or as a scheduled task. Continue with the following steps.

Execute Immediate (F8)

1. Choose *Execute* or F8.
2. The system displays a pop-up message advising you that the data for the selected function will be deleted. It asks you for confirmation to proceed:

- If you choose *Yes*, the data is deleted.
- If you choose *No*, the system cancels the program.

Execute in Background – Immediate

1. Choose *Program* > *Execute in Background* on the main menu.
2. On the *Background Print Parameters* screen, choose *Continue*.
3. On the *Start Time* screen, choose *Immediate* and *Save* consecutively to run the program in the background.
4. To view the progress of the background run, launch transaction code **SM37**.

Execute in Background – Scheduled

1. Choose *Program* > *Execute in Background* on the main menu.
2. On the *Background Print Parameters* screen, choose *Continue*.
3. On the *Start Time* screen, choose *Date/Time*.
4. The *Date/Time* section appears, where you can specify the date and time that you want to schedule for the run.
5. To view the progress of the background run, launch transaction code **SM37**.

1.4 Functions

Financial and business models consist of functions that are connected to each other by means of input-output relationships.

The output of one function can be the input of multiple other functions, and in this way complex calculations and logic can be modeled in a comfortable way.

The following functions are available:

Key Feature	Use
Allocation [page 139]	Performs direct and indirect allocations
Calculation [page 150]	Performs mathematical formulas
Calculation Unit [page 337]	Encapsulates a group of functions and makes them reusable
Condition [page 178]	Defines a condition which acts as a trigger for the processing logic of the child functions
Conversion [page 181]	Performs currency and unit conversions
Derivation [page 187]	Performs if-then-else enrichments of data
Description [page 346]	Describes processes and topics used for the documentation of models

Key Feature	Use
File Adapter The behavior of this function depends on the chosen file IO type: • Import [page 109] • Export [page 303]	Provides automated access to files
Flow Modeling [page 191]	Provides calculation for the best-estimate cash flow (BECF)
Funds Transfer Pricing [page 253]	Performs funds and liquidity transfer pricing calculations
Join [page 267]	Performs collections, joins, unions and lookups for separate data
Machine Learning [page 362]	Provides a rule type to train and uses a time-series forecast model based on input data
Model BW [page 114]	Provides read and write access to a local BW InfoSource like Advanced DSOs
Model Join [page 117]	Performs collections, joins, unions and lookups for separate data just like the *Join* function, but as a data model function.
Model RDL [page 131]	Provides read and write access to a local FRDP Results Data layer
Model Table [page 134]	Provides read and write access to a local or remote data table
Model View [page 137]	Provides read access to a local or remote data table or view
Query [page 347]	Allows the output and input of data
Remote Function Adapter [page 305]	Performs an ABAP-based remote function call (for example, a call to a remote FI-GL posting BAPI)
Transfer Structure [page 282]	Performs a transfer from accounting-based data to costing-based data (also called “denormalization”)
Valuation [page 289]	Performs comprehensive calculations with different valuation methods (for example, discounting)
View [page 294]	Projects or aggregates data, including filtering options and formulas
Writer [page 323]	Stores data in a model table, Model RDL or model BW

Related Information

For more information about common aspects of SAP Profitability and Performance Management functions, see Concepts for Key Users [page 6].

For more information about building functions in SAP Profitability and Performance Management, see How to Build SAP PaPM Functions.

1.4.1 Information Functions

These functions define the data and information model in an environment. Technically, they act as a proxy that contains the details required to read and – if allowed – write data from and to this data model. Functionally, they define or display the available fields from that model.

The following information functions are provided:

- File Adapter (Type "Import") [page 109]

  > **i** Note
  >
  > There is also a *File Adapter* function of type “Export” (see here [page 303]), which belongs to another function category (Write and Adapter Functions [page 302]).

- Model BW [page 114]
- Model Join [page 117]
- Model RDL [page 131]
- Model Table [page 134]
- Model View [page 137]

1.4.1.1 File Adapter (Type "Import")

The *File Adapter* function of type “Import” provides automated access to files so that file content can be imported as input for calculations.

Key Features

Header

In the header, you define the principal behavior of the *File Adapter* function. The following fields are available:

- *File IO Type*: Choose “Import” to import data from a server file.
- *File Format*: Refers to the definition of a file format, which is maintained centrally for the environment.
- *File Name*: Specifies the name of the file on the server that is used. Use the following format: “<File name>.ext”.

> **Note**
>
> - Period (.) is not supported as a character in the file name (for example, do not use a file name like “importfile.1.csv”).
> - If a parameter is used instead of the actual name of the file, the function must be activated with the file name first, then the parameter can be used.

- *Header Row*: Defines the row number in which the header columns are available. The value 0 means that there is no header row.
- *Number of Threads*: Multiple threads can reduce the import time. The maximum permitted value is 255.
- *Batch Size*: Specifies the number of records to be inserted in each commit.
- *Table Lock*: If this is set, the data for column store tables is loaded faster.
- *No Type Check*: Specifies that the records are inserted without checking each field type.
- *Fail on Invalid Data*: Specifies that the import fails unless all the entries are imported without errors.
  - If you mark this checkbox in the header of the function, the system will display any exceptions that may occur during run time on the UI.
  - If this checkbox is not marked and no exceptions occur during run time, the system will display a message to inform you that a log file has been created. This is only the case if the displayed data or behavior is not as expected.

Server Files

This is a helper tab that has no influence on the runtime of the function.

The *Refresh Directory List* button shows a list of files that are currently available on the server. The content of these files can also be viewed here. Use the *Select File* button to register it in the header as the file name to be used.

Small files can also be uploaded and downloaded but we recommend that you use server-side IT-driven mechanisms to manage files in the server directory.

Preview

This is a helper tab that has no influence on the runtime of the function.

Once the file name is set in the header, the *Preview* button allows you to preview the file.

Stage

This is a helper tab that has no influence on the runtime of the function.

Once the file name is set in the header, the *Stage* tab allows you to stage the file in a temporary table separating the data into columns. This makes it easier to analyze data, including filtering, sorting, and checking.

Mapping

The file columns can be mapped to existing fields in the environment. The *Field Mapping Proposal* button helps you to match columns to field names.

If the used file has a header row (a row containing the column names), the file columns mapped on the *Mapping* tab must have the same name as the columns from the file.

Optionally, formulas can be defined to convert data.

Procedure

> **Note**
>
> Before you can use the *File Adapter* function, you have to configure the *File Formats* section on the *Environment Details* screen. For more information about this section, see File Formats [page 60].

Function Access

Follow the steps below to access the *File Adapter* function of type "Import":

1. In the client where SAP Profitability and Performance Management is installed, choose *SAP Menu* > *Profitability and Performance Management* > *Modeling* > *Start My Environments*.
2. The *Environment* screen appears in a separate browser window. Choose an existing environment and continue. Within the environment, you can set up the newly added function.

Function Configuration

Follow the steps below to configure the *File Adapter* function of type "Import":

1. In edit mode, configure the following required fields in the header. For more information about the header, see the *Key Features* section above.
   - *File IO Type* = "Import"
   - *File Format*: Choose the file format configured in *Environment* > *File Formats*
   - *File Name*
   - *Header Row*
   - *Number of Threads*
   - *Batch Size*
   - *Table Lock*
   - *No Type Check*
   - *Fail on Invalid Date*
2. Define the fields to be used on the *Signature* tab.

   > **Note**
   >
   > For more information about the *Signature* tab, see Signature [page 13].
3. Choose *Field Mapping Proposal* on the *Mapping* tab to generate the mapping of columns to field names with field attributes.
4. Define the checks to be used on the *Checks* tab, if needed.

   > **Note**
   >
   > For more information about the *Checks* tab, see Checks [page 17].
5. Choose *Save* and then choose *Activate*.

Related Information

- For more information about common aspects of SAP Profitability and Performance Management functions, see Concepts for Key Users [page 6].
- For more information on how to add and remove functions in SAP Profitability and Performance Management environments, see Function Hierarchy [page 63].
- For more information about using parameters in the File Adapter function, see Profitability and Performance Management (PaPM) – Using Parameter in the File Adapter Function

1.4.1.1.1 Example: Importing CSV Files

In this example, we import a CSV file from a file server to SAP Profitability and Performance Management.

> **i** Note
>
> It is important that you set the file IO type to “IMPORT”.

Input Data

ZE_CUST (Customer)	ZE_PROD (Product)	ZX_QTY (Quantity)	ZE_AMT (Amount)
CN002	PROD01	40	23
CN001	PROD01	20	20
CN004	PROD03	30	33
CN003	PROD03	30	25
CN001	PROD02	33	40
CN004	PROD01	25	28
CN002	PROD02	25	30

Procedure

Uploading a CSV to the file server

Convert the input data above into a CSV file format and upload it to the server /usr/sap/trans70/hana (the whitelist path).

You can configure the whitelist path under *Environment* > *File Format Tab*.

Server File Tab

1. In the function details, while in edit mode, choose *Upload*.
2. Choose the input data that you want to upload. Select the file name of the CSV file.

> **i** Note
>
> This is the CSV file created from the data in the *Input Data* section.

Use the CSV file in the function

1. Once you have uploaded the file to the server, choose the *Refresh Directory List* button.
2. Select .csv file, then choose the *Select File* button.
3. The system displays the file name in the header section as " <File name>.csv".

Mapping of Fields

1. You can map the file columnsto existing fields in the environment. In edit mode, choose the *Field Mapping Proposal* button. This button helps to match columns to field names.
2. In the *File* column, there should be a field named `ZX_QTY` (Quantity).
3. Map `ZX_QTY` to `ZE_QTY` by assigning `ZE_QTY` in the *Field* column.

Expected Result

> **i** Note
>
> *File Adapter* is an integration function to consume text file (CSV) records that are used as input for SAP Profitability and Performance Management. The system does not carry out any additional processing in this scenario except for reading data from the Excel file and preparing it for consumption by the processing functions, like *Allocation* or *Join*.

The result provided by the function can be used either as input for other functions or for calculation. In this example, the result must be structured as shown in the table below.

> **i** Note
>
> Now `ZE_QTY` is stated instead of `ZX_QTY` since this is the configuration done on the mapping tab.

ZE_CUST (Customer)	ZE_PROD (Product)	ZE_QTY (Quantity)	ZE_AMT (Amount)
CN002	PROD01	40	23
CN001	PROD01	20	20
CN004	PROD03	30	33
CN003	PROD03	30	25
CN001	PROD02	33	40
CN004	PROD01	25	28

ZE_CUST (Customer)	ZE_PROD (Product)	ZE_QTY (Quantity)	ZE_AMT (Amount)
CN002	PROD02	25	30

1.4.1.2 Model BW

Model BW is a data model function that allows you to define Advanced Data Store Objects (ADSO) in the SAP Profitability and Performance Management environment or to access external BW InfoProviders defined in SAP Business Warehouse (SAP-BW).

Key Features

Header

In the header, you can choose any of the following connection types:

- If you select the connection type “Logical”, only the field *Connection Name* is available for selection.
- If you select the default connection type “Physical” the following Model BW sources are available for selection:
  - Business Warehouse [page 114]
  - Environment [page 116]

The function provides options based on the Model BW source that you have chosen.

1.4.1.2.1 Model BW Source: Business Warehouse

The data model is defined and managed externally in SAP Business Warehouse. This function allows the BW InfoProviders to be referenced and made available in the environment.

Key Features

Header

You can choose the appropriate BW InfoProvider. Ensure that the BW InfoProvider that you choose has an external SAP HANA view generated in SAP-BW.

> **i Note**
>
> You can search for BW InfoProviders by technical name, short description and long description.

Fields

When you choose *Synchronize*, the system automatically reads the external SAP HANA view for the BW InfoProvider and makes field proposals in the *Sync Model Fields* dialog. The field state denotes whether the fields are already available or new fields have been added to the environment. You can change the field name and description of the new fields. However, you need to ensure that the name is unique. You can also exclude certain fields from read access. If you want the changes to be reset to the initial state, choose *Reset Proposal*.

Parameters

If input parameters have been defined for the generated external SAP HANA view, the parameters appear here, and a constant value or an environment parameter can be assigned.

Procedure

Function Access

Follow the steps below to access the *Model BW* function with source "Business Warehouse":

1. In the client where SAP Profitability and Performance Management is installed, choose *SAP Menu* > *Profitability and Performance Management* > *Modeling* > *Start My Environments*.
2. The *Environment* screen appears in a separate browser window. Choose an existing environment and continue. Within the environment, you can set up the newly added function.

Function Configuration

Follow the steps below to configure the *Model BW* function with source "Business Warehouse":

1. In edit mode, configure the following required fields in the header. For more information about the header, see the *Key Features* section above.
   - *Connection Type*: Choose either "Physical" or "Logical".
   - *Model BW Source* = "Business Warehouse"
   - *BW Info Provider*
2. Choose *Synchronize* on the *Fields* tab.
3. The *Sync Model Fields* dialog appears showing the fields to be imported from the BW source.
4. Choose the fields to be added, then choose *OK*.
5. Define the parameters on the *Parameters* tab, if needed.
6. Choose *Save* and then choose *Activate*.

Related Information

- For more information about common aspects of SAP Profitability and Performance Management functions, see Concepts for Key Users [page 6].
- For more information on how to add and remove functions in SAP Profitability and Performance Management environments, see Function Hierarchy [page 63].

1.4.1.2.2 Model BW Source: Environment

The data model is defined and managed in the environment.

Key Features

Header

You can make the following settings in the *Editable* field :

- If you choose “Yes”, the system allows manual data input or planning.
- If you choose“No”, the system does not allow manual data input or planning.

During activation, the system automatically generates Advanced Data Store Objects (ADSO) of the type “direct update-planning”. You can find the InfoProvider name under *Function Runtime Attributes*. All characteristic and unit fields are set as keys. The system determines all the required settings; you cannot make settings yourself or change these settings. The system also generates the external SAP HANA view of the ADSO.

Fields

On the *Fields* tab, you can choose and add fields or InfoObjects from the field list to the Model BW. Ensure that you first include new fields or InfoObjects in the environment to make them available in the field list.

If navigational attributes have been defined for the selected InfoObject, you can select the *Navigational Attribute on*checkbox. The fields marked in this way appear as a new column when the Model BW is used in the input function.

Parameters

If input parameters have been defined for the generated external SAP HANA view, the parameters appear here, and a constant value or an environment parameter can be assigned.

Procedure

Function Access

Follow the steps below to access the *Model BW* function with source “Environment”:

1. In the client where SAP Profitability and Performance Management is installed, choose *SAP Menu* > *Profitability and Performance Management* > *Modeling* > *Start My Environments* .
2. The *Environment* screen appears in a separate browser window. Choose an existing environment and continue. Within the environment, you can set up the newly added function.

Function Configuration

Follow the steps below to configure the *Model BW* function with source “Environment”:

1. In edit mode, configure the following required fields in the header. For more information about the header, see the *Key Features* section above.

- *Connection Type*: Choose either “Physical” or “Logical”.
- *Model BW Source* = “Environment”
- *Editable*

2. On the *Fields* tab, choose + (*Add*).
3. The *Add Fields* screen appears. Choose the fields to be added, then choose *OK*.
4. Define the parameters on the *Parameters* tab, if needed.
5. Choose *Save* and then choose *Activate*.

Related Information

- For more information about common aspects of SAP Profitability and Performance Management functions, see Concepts for Key Users [page 6].
- For more information on how to add and remove functions in SAP Profitability and Performance Management environments, see Function Hierarchy [page 63].

1.4.1.3 Model Join

The *Model Join* function type is a data model function that, just like the *Join* function, involves possible complex selections and the enrichment of one or more data sources.

Aside from the differences regarding the technical and buffering mechanisms, modeling of these two functions in the environment is performed in a similar manner with only slight differences.

From a technical perspective, *Join* is a processing function whereas *Model Join* is subordinated to the information functions. This means upon successful activation of a Model Join a SQL view is generated in the underlying database schema of the tenant. Thus, the *Model Join* function type behaves similar to an information function: no procedure is created; the function is purely projecting the content of the tables or views. The *Join* processing function on the other hand generates SAP HANA procedures that are then being executed to populate a temporary Ytable.

Regarding modeling, *Model Join* focuses on simplicity and does not have neither *Input* nor *Checks* nodes. Partitioning is not possible. Setting the function to “Executable” and changing the event handling aside from logging is not allowed. The *Model Join* function also does not provide the usual join features intended for processing functions (for example, *Include Original Input Data*, *Processing Type*, and so on). Most importantly, unlike the *Join* function which can have any other processing functions as input, the *Model Join* function can only use other information functions such as *Model Table* and *Model View*.

Model Join is also behaving as a normal *Model View* function, which in the end provides all data necessary and does not filter out null values.

To use the processing features as mentioned, a processing function (for example *View*) can be used on top before feeding the result to the *Query* function.

Key Features

Header

In the header, you define the principal behavior of the Model Join.

You can choose between different Model Join types:

- “Implicit Fields”: Fields coming from inputs are automatically considered during the join procedure. If the field is visible on multiple join rules (or inputs), you can use the *Auto Filling* function.
- “Explicit Fields”: Fields of inputs are manually defined and maintained by the modeler. These fields will then be considered during processing.

> **Example**
>
> In the “Union All” join type, you need to add all fields required for the output even if they are not part of the input data. The fields must be added on each subview of the rules to be joined before activation.

You can control the Model Join results with the *Auto Filling* option. The following settings are possible:

- Do not use the option “No” for the *Auto Filling* field, because *Model Join* also behaves as a normal Model View which in the end provides all required data and does not filter out null values.
- “If Null then First to Last”: The first non-null value is taken, and if all values are null, the initial value is returned for that field.
- “If Null/Initial then First to Last”: The first non-null and non-initial value is taken, and if all values are null or initial, an initial value is returned for that field.

> **Note**
>
> **Null vs. Initial**
>
> On the SAP Profitability and Performance Management Cloud UI, both NULL and INITIAL are represented by an empty cell for the characteristic and “0” for the key figure. The modeler must take care when processing input data with these values because the response is different:
>
> - NULL (?) does not have any value and does not occupy memory. This can be the case if a field was not assigned with values initially.
> - INITIAL (' ') has a value and therefore occupies memory. In SAP Profitability and Performance Management Cloud, a cell can have an INITIAL value if it was assigned an empty record.

Rules

Each Model Join rule semantically defines the reading of a specific input.

Hierarchical Model Join rules are also supported by assigning higher levels. The hierarchy of levels is resolved starting with the highest level, feeding as input to the lower levels, and ending with level 0.

The following rule types are available:

- “From”: This is always the first rule of a level.
- “Left Outer Model Join”: Returns all rows from the rule above as well as the columns and rows from this rule that match the predicates.
- “Inner Model Join”: Returns all rows when there is at least one predicate match in the rule above and this rule.

- “Full Outer Model Join”: Returns all (matched or unmatched) rows from both the rule above and this rule.
- “Cross Model Join”: Returns the cartesian product of the rule above and this rule.
- “Union All”: Behaves in the same way as a union, but duplicate records are not removed.
- “Lookup”: Looks up fields and fills them in the first non-lookup rule above where the predicates match. At least one field needs to be defined as a lookup field.
- “Lookup Auto Predicate”: Looks up fields and fills them in the first non-lookup rule where all common fields match those of the input function which is set on the *Rule* tab (this means, Field 1 of Rule 1 = Field 1 of Rule 2). At least one field needs to be defined as a lookup field.

You can also configure the following Model Join rule settings, if required:

- *Sub View*: You can define further selections, formulas, aggregations and sorting orders for each rule.
- *Complex Selections*: You can define complex selections using formulas and SQL functions.
- *Join Predicates*: You can define the predicate conditions for the matching for Model Join and lookup rules here.
- *Complex Predicates*: You can enter complex on-predicates for Model Join and lookup rules here using formulas and SQL functions.

> **i Note**
>
> Fields coming from inputs are automatically considered during the Model Join procedure. If the field is visible, these settings are only relevant for multiple Model Join rules in which either non-null or non-initial values of the same field are considered and returned for that field. If there is only one rule, the *Auto Filling* options are not relevant.
>
> Using SAP HANA hints in complex selections can help in scenarios where a Model Join function execution runs into an out-of-memory exception or is very slow. For more information about which hint could help in your implementation-specific scenario (rules at the same level or multi-level rules that lead to subqueries, formula/selection being used), see the SAP HANA Help for HINT.

Procedure

Function Access

Follow the steps below to access the *Model Join* function:

1. In the client where SAP Profitability and Performance Management is installed, choose *SAP Menu* > *Profitability and Performance Management* > *Modeling* > *Start My Environments*.
2. The *Environment* screen appears in a separate browser window. Choose an existing environment and continue. Within the environment, you can set up the newly added function.

Function Configuration

Follow the steps below to configure the *Model Join* function:

1. In edit mode, configure the required fields *Join Type* and *Auto Filling* in the header. For more information about the header, see the *Key Features* section above.
2. Define the fields to be used on the *Signature* tab.

> **i Note**
>
> For more information about the *Signature* tab, see Signature [page 13].

3. On the *Rules* tab, you can define the rule types of the *Model Join* function.
   Proceed as follows to configure the rules:
   1. On the *Rules* tab, choose + (*Add*).
   2. The *Add Details* screen is displayed. Enter the following information:
      - *Add Level*
      - *Rule*
      - *Rule Type*
      - *Input Description*
   3. Choose *OK*.
   4. Select the created rule.
   5. Define the input function to be used.
   6. On the *Subview* tab of the created rule, choose + (*Add*).
   7. The *Fields* screen appears. Select the fields to be used and choose *OK*.
   8. Follow the steps below if you want to setup or apply a formula for a specific field (optional):
      1. Select the field and choose *f* (*Formula*).
      2. On the *Formula* screen which is displayed next, enter the formula.
      3. Choose *OK*.
   9. Follow the steps below if you want to setup or apply a selection condition for a specific field (optional):
      1. Select the field and choose *Selection Condition*.
      2. On the *Select Condition* screen which is displayed next, enter the condition.
      3. Choose *OK*.
   10. On the *Complex Selections* tab of the created rule, you can directly define complex selections using formulas and SQL functions.
   11. On the *Join Predicates* tab of the created rule, choose + (*Add*).
   12. The system automatically adds a new line entry. Enter the following information here:
      - *Field*
      - *Comparison*
      - *Join Rule*
      - *Join Field*
   13. On the *Complex Predicates* tab of the created rule, you can directly enter “on” predicates for Model Join and lookup rules using formulas and SQL functions.
4. Choose *Save* and then choose *Activate*.

Related Information

For more information about common aspects of SAP Profitability and Performance Management functions, see Concepts for Key Users [page 6].

For more information on how to add and remove functions in SAP Profitability and Performance Management environments, see Function Hierarchy [page 63].

1.4.1.3.1 Full Outer Model Join Special Scenarios for Auto Filling Field

1.4.1.3.1.1 Example: Full Outer Model Join – Auto Filling Set to "No"

This scenario shows how the rule type “Full Outer Model Join” merges two input tables with one join predicate as a bridge.

Input Tables

Product - Material Table

Product Code	Material Code	Request Order
P0001	M1011	5
P0001	M1010	2
P0002	M1009	1
P0002	M1011	3
P0005	M1012	10
P0005	M1011	30

JO - Product Table

Product Code	Product	Price	Unit
P0001	Shoe	75	EUR
P0002	Watch	300	EUR
P0003	Shirt	80	EUR
P0004	Shorts	20	EUR

This scenario executes a Full Outer Model Join that is based on the join predicate.

In this case, *Product Code* is the predicate used for both rules and all items with product code “P0001” and “P0002” are added in the final results.

Interim Result (Product - Material Table and JO - Product Table)

Product - Material Table

Product Code	Material Code	Request Order
P0001	M1011	5
P0001	M1010	2
P0002	M1009	1
P0002	M1011	3

JO - Product Table

Product Code	Product	Price	Unit
P0001	Shoe	75	EUR
P0002	Watch	300	EUR

The following table shows parts of the Full Outer Join table. However, since they contain “?” or null values, they are not included in the final results.

Product Code	Material Code	Request Order
P0005	M1012	10
P0005	M1011	30
	?	?
	?	?

Product Code	Product	Price	Unit
	?	?	?
	?	?	?
P0003	Shirt	80	EUR
P0004	Shorts	20	EUR

As mentioned above, the system returns the rest of the non-null and/or non-initial values since the *Model Join* function is also behaving as a normal *Model View* function.

Expected Result

Material Codw	Request Order	Product Code	Product	Price	Unit
M1011	5	P0001	Shoe	75	EUR
M1010	2	P0001	Shoe	75	EUR
M1009	1	P0002	Watch	300	EUR
M1011	3	P0002	Watch	300	EUR
M1011	30	P0005		0	
M1012	10	P0005		0	
			Shirt	80	EUR
			Shorts	20	EUR

Returns values for matching rows (“P0001” and “P0002”) based on the join predicates set for the field *Product Code* (`PROD_CODE`).

1.4.1.3.1.2 Example: If Null/Initial Then First to Last

This scenario shows the enrichment capability of the *Model Join* function to highlight the effect of *Auto Filling* set to “If Null/Initial then First to Last”.

That means, the system substitutes null and initial values (for *Characteristic* it is whitespace (' ') and for *Key Figure* it is zero ('0')) by the next non-null value of the succeeding table having the same field. If this field only has initial/null values, the system just sets an initial value.

Input Tables

Legend: '' is considered as an initial or empty input

JO – Product / Customer in US

Product	Customer	Amount
PROD01	US_CUST01	200
PROD04	US-CUST04	100
PROD06	US_CUST06	300
PROD07	''	100
PROD08	''	200
PROD09	US_CUST09	300

JO – Product / Customer in DE

Product	Customer	Price
PROD01	DE_CUST01	120
PROD04	DE_CUST04	60
PROD05	DE_CUST05	180
PROD07	DE_CUST07	60
PROD08	DE_CUST08	120
PROD10	DE_CUST10	180

The system takes the first non-null and non-initial value and if all values are null or initial, it returns an initialized value, empty or blank for a Character (CHAR) field and "0" for a *Key Figure* field.

Interim Results

Product	Customer	Amount	Price
PROD01	US_CUST01	200	120
PROD04	US_CUST04	100	60
PROD05	DE_CUST05	?	180
PROD06	US_CUST06	300	?
PROD07	''	100	60
PROD08	''	200	120
PROD09	US_CUST09	300	?
PROD10	DE_CUST10	?	180

In PROD05 row, we had our first null value (?). Since we only have two tables we won't be able to look further for another initial value. Since all values are null for the *Amount* field, the system returns an initialized value. For *Key Figure*, it will be "0". The same scenario will be encountered for PROD06, PROD09 and PROD10.

In PROD07 row, we had our first initial value (''), as you can see we look at the next table for the same *Customer* field. Since the next value for Customer is "DE_CUST07", it will be the value for *Customer* field at PROD07. The same is true for PROD08 scenario, which will have a value of "DE_CUST08".

Expected Result

Product	Customer	Amount	Price
PROD01	US_CUST01	200	120
PROD04	US_CUST04	100	60
PROD05	DE_CUST05	0	180
PROD06	US_CUST06	300	0
PROD07	DE_CUST07	100	60
PROD08	DE_CUST08	200	120
PROD09	US_CUST09	300	0

Product	Customer	Amount	Price
PROD10	DE_CUST10	0	180

1.4.1.3.1.3 Example: If Null Then First to Last

This scenario shows the enrichment capability of the *Model Join* function to highlight the effect of *Auto Filling* set to “If Null then First to Last”.

That means, the system substitues null values (' ') of the fields by the next non-null value of the succeeding table having the same field. If this field only has null values, the initial value is set.

Input Tables

Legend: " is considered as an initial or empty input

JO – Product / Customer in US

Product	Customer	Amount
PROD01	US_CUST01	200
PROD04	US-CUST04	100
PROD06	US_CUST06	300
PROD07	"	100
PROD08	"	200
PROD09	US_CUST09	300

JO – Product / Customer in DE

Product	Customer	Price
PROD01	DE_CUST01	120
PROD04	DE_CUST04	60
PROD05	DE_CUST05	180
PROD07	DE_CUST07	60
PROD08	DE_CUST08	120
PROD10	DE_CUST10	180

The system takes the first non-null value and if all values are null, it returns the initial value for that field.

Interim Results

Product	Customer	Amount	Price
PROD01	US_CUST01	200	120
PROD04	US_CUST04	100	60
PROD05	DE_CUST05	?	180
PROD06	US_CUST06	300	?
PROD07	"	100	60
PROD08	"	200	120
PROD09	US_CUST09	300	?
PROD10	DE_CUST10	?	180

In the row PROD05, we have our first null value (?). Since all values are null for the *Amount* field, the system returns an initialized value. For *Key Figure*, the value is 0. The same scenario applies to PROD06, PROD09 and PROD10.

In the row PROD07, we have our first initial value (''). Since this initial value ('') is a non-null value, it is used as the result. The same is true for the scenario PROD08, which has a value of empty or blank because the type is “Character”.

Expected Result

Product	Customer	Amount	Price
PROD01	US_CUST01	200	120
PROD04	US_CUST04	100	60
PROD05	DE_CUST05	0	180
PROD06	US_CUST06	300	0
PROD07		100	60
PROD08		200	120
PROD09	US_CUST09	300	0
PROD10	DE_CUST10	0	180

1.4.1.3.2 Example: Inner Model Join

This scenario shows how the inner Model Join merges the two input tables and returns values that match with the set Model Join predicate.

Input Tables

Product - Material Table

Product Code	Material Code	Request Order
P0001	M1011	5
P0001	M1010	2
P0002	M1009	1
P0002	M1011	3
P0005	M1012	10
P0005	M1011	30

Product Table

Product Code	Product	Price	Currency
P0001	Shoe	75	EUR
P0002	Watch	300	EUR
P0003	Shirt	80	EUR
P0004	Shorts	20	EUR

Returns all values with corresponding matches based on the join predicates for product code “P0001” and “P0002”.

Product, *Price* and *Currency* which correspond to each *Product Code* will be distributed to each product code resulting in the below table.

Interim Results (Product - Material Table and Product Table)

Product Code	Material Code	Request Order	Product	Price	Currency
P0001	M1011	5	Shoe	75	EUR
P0001	M1010	2	Shoe	75	EUR

Product Code	Material Code	Request Order	Product	Price	Currency
P0002	M1009	1	Watch	300	EUR
P0002	M1011	3	Watch	300	EUR

Since a formula has been declared in the subview of RULE2, SAP Profitability and Performance Management will perform it after the Inner Join has been run.

The system adds the field *Payment Amount* (Payment Amount = Request Order * Price) along with its value to the final output.

Expected Result

Product Code	Material Code	Request Order	Product	Price	Currency	Payment Amount
P0001	M1011	5	Shoe	75	EUR	375
P0001	M1010	2	Shoe	75	EUR	150
P0002	M1009	1	Watch	300	EUR	300
P0002	M1011	3	Watch	300	EUR	900

1.4.1.3.3 Example: Left Outer Model Join

This scenario shows how the left outer Model Join merges the two input tables and returns values that match with the set join predicate.

Input Tables

Product - Material Table

Product Code	Material Code	Request Order
P0001	M1011	5
P0001	M1010	2
P0002	M1009	1
P0002	M1011	3
P0005	M1012	10
P0005	M1011	30

Product Table

Product Code	Product	Price	Currency
P0001	Shoe	75	EUR
P0002	Watch	300	EUR
P0003	Shirt	80	EUR
P0004	Shorts	20	EUR

Returns all values with corresponding matches based on the join predicates for product codes P0001 and P0002.

The product codes P0005, P0003 and P004 have no matches in tables. They will be dropped in the result.

Expected Result

Product Code	Material Code	Request Order	Product	Price	Currency
P0001	M1011	5	Shoe	75	EUR

Product Code	Material Code	Request Order	Product	Price	Currency
P0001	M1010	2	Shoe	75	EUR
P0002	M1009	1	Watch	300	EUR
P0002	M1011	3	Watch	300	EUR

Returns all values from the left table and all corresponding matches based on the join predicates for product codes P0001 and P0002.

1.4.1.3.4 Example: Cross Model Join Implicit

This scenario merges the three input tables and returns all row combinations from the records of the inputs.

Input Tables

Material Table

Material Code	Material	Cost per Order	Unit
M1011	Leather	10	USD
M1012	Thread	5	USD

Product Table

Product Code	Product
P0001	Shoe
P0002	Watch

Order Table

Branch	Order
BR001	10
BR002	20

Interim Result of Material Table and Product Table (MatPro Table)

Material Code	Material	Cost per Order	Unit	Product Code	Product
M1011	Leather	10	USD	P0001	Shoe
M1012	Thread	5	USD	P0001	Shoe
M1011	Leather	10	USD	P0002	Watch
M1012	Thread	5	USD	P0002	Watch

Returns material code M1011 and M1012 for product code P0001.

Returns material code M1011 and M1012 for product code P0002.

The Cross Join produces an interim result set which is the number of rows in the first table multiplied by the number of rows in the second table.

Expected Result

Material Code	Material	Cost per Order	Unit	Product Code	Product	Branch	Order
M1011	Leather	10	USD	P0001	Shoe	BR001	10
M1011	Leather	10	USD	P0001	Shoe	BR002	20
M1011	Leather	10	USD	P0002	Watch	BR001	10
M1011	Leather	10	USD	P0002	Watch	BR002	20

Material Code	Material	Cost per Order	Unit	Product Code	Product	Branch	Order
M1012	Thread	5	USD	P0001	Shoe	BR001	10
M1012	Thread	5	USD	P0001	Shoe	BR002	20
M1012	Thread	5	USD	P0001	Watch	BR001	10
M1012	Thread	5	USD	P0002	Watch	BR002	20

The Cross Model Join produces a result set which is the number of rows in the interim table result multiplied by the number of rows on the third table.

In this result, we are identifying the number of orders, per product and per branch by cross referencing from *Material Table* to *Product Table* and *Order Table*.

1.4.1.3.5 Example: Union All

This scenario shows how the *Model Join* function merges the two input tables based on a set of rules.

Input Tables

JO - Material Table

Material Code	Material	Cost per Order	Currency
M1001	Paper	3	USD
M1002	Plastic	3	USD
M1003	Wax	4	USD
M1006	Botton	1	USD
M1007	Cotton	10	USD
M1009	Glass	50	USD
M1010	Lace	5	USD
M1011	Leather	10	USD
M1012	Thread	5	USD

JO - Product Table

Product Code	Product	Price	Currency
P0001	Shoe	75	EUR
P0002	Watch	300	EUR
P0003	Shirt	80	EUR
P0004	Shorts	20	EUR

Filter the tables first so that you have only the required entries:

- For the Material Table, we set the condition to M1001, M1002 and M1003.
- For the Product Table, we set the condition to P0001, P0002 and P0003.

Based on the selection conditions we set in the subview of each rule, we will have the following tables:

Interim Result

Material Table

Material Code	Material	Cost per Order	Currency
M1001	Paper	3.00	USD
M1002	Plastic	3.00	USD
M1003	Wax	4.00	USD

Product Table

Product Code	Product	Price	Currency
P0001	Shoe	75	EUR
P0002	Watch	300	EUR
P0003	Shirt	80	EUR

If we are using an explicit type of join, the fields that we define in the subview of each rule will be the output. However, we need to specify all the fields that we need in the subview of each rule because we need to have the same fields across the subview for the explicit view to work.

Union All is used to combine the result sets of two or more tables. It does not remove duplicate rows and all rows are returned.

Expected Result

Material Code	Material	Product Code	Product	Cost per Order	Price	Currency
M1001	Paper			3	0	USD
M1002	Plastic			3	0	USD
M1003	Wax			4	0	USD
		P0001	Shoe	0	75	EUR
		P0002	Watch	0	300	EUR
		P0003	Shirt	0	80	EUR

1.4.1.3.6 Example: Lookup Auto Predicates

For this scenario, corresponding material is retrieved and displayed for every matching entry in the field *Material Code*.

Level 1 Processing

Product - Material Table will be enriched as the corresponding *Material* will be retrieved and displayed for every matching entry in the field *Material Code* (`MAT_CODE`).

> **i** Note
>
> The system resolves the hierarchy of levels starting with the highest level, feeding as an input to the lower levels and ending with level 0.

Product - Material Table (Level 1, From)

Product Code	Material Code	# Request Order
P0001	M1011	5
P0001	M1010	2
P0002	M1009	1
P0002	M1011	3
P0005	M1012	10
P0006	M1011	30

Material Table (Level 1, Lookup Auto Predicate)

Material Code	Material	Cost per Order	Unit
M1001	Paper	3	USD
M1002	Plastic	3	USD
M1003	Wax	4	USD
M1006	Botton	1	USD
M1007	Cotton	10	USD
M1009	Glass	50	USD
M1010	Lace	5	USD
M1011	Leather	10	USD
M1012	Thread	5	USD

Interim Result (Level 1)

Product Code	Materials Code	# Request Order	Material
P0001	M1011	5	Leather
P0001	M1010	2	Lace
P0002	M1009	1	Glass
P0002	M1011	3	Leather
P0005	M1012	10	Thread
P0006	M1011	30	Leather

Level 0 Processing

Level 0 Processing

The product table declared in the first rule (“From”) will now perform a Left Outer Model Join (for every matched product code (`PROD_CODE`) entry) with the Level 1 result (enhanced Product - Material Table) since result processed from a higher level will be considered as an input for the lower level.

> **i** Note
>
> Setting the Product - Material Table as an input function for the second rule will not affect the results of the join since the system automatically detects that the input will be coming from the enhanced Product - Material Table.

Product Table (Level 0, From)

Product Code	Product	Price	Unit
P0001	Shoe	75	EUR
P0002	Watch	300	EUR
P0003	Shirt	80	EUR
P0004	Shorts	20	EUR

Interim Result (Level 1)

Product Code	Materials Code	# Request Order	Material
P0001	M1011	5	Leather
P0001	M1010	2	Lace
P0002	M1009	1	Glass
P0002	M1011	3	Leather
P0005	M1012	10	Thread
P0006	M1011	30	Leather

Expected Result

Product Code	Product	Price	Unit	Material Code	Request Order	Material
P0001	Shoe	75	EUR	M1011	5	Leather
P0001	Shoe	75	EUR	M1010	2	Lace
P0002	Watch	300	EUR	M1009	1	Glass
P0002	Watch	300	EUR	M1011	3	Leather

1.4.1.4 Model RDL

Model RDL is a data model function that allows you to access SAP HANA-optimized Results Data types of the Financial Products subledger datamodel.

Key Features

Header

Header fields are important to be filled correctly in order to connect to an existing and activated data model. To perform this, you have to define the connection type, results data area and the result type.

> **Example**
>
> Result Data Area: `SRINS`
>
> Result Type: `S_ACG`

You must also define the view type, either “Application” or “Integration”. For Application View, the field names format is `/BA1/*` and for Integration View, the fields names format is `BA1_*`.

Fields

While on edit mode, choose the *Synchronize* button. The system then displays a popup window allowing you to exclude fields by setting the *Exclude* checkbox to marked (checked). All excluded fields will not be considered during processing in SAP Profitability and Performance Management.

Simple Reading Mechanism, Selection using Fixed Value Model RDL (for example View, Join)

SAP Profitability and Performance Management reads data records from a specific Result Type through active and generated SQL or HANA Views. The system automatically consumes all data records written in the HANA view.

In order to filter results with fixed selection, you have to create an SAP Profitability and Performance Management processing function such as the *View* or *Join* function on top of the Model RDL.

To perform this, follow below steps:

1. Create a processing function such as *View* with the Model RDL as the input function.
2. Select this *View* function and on the right side of the screen and choose *Edit*.
3. Add a selection by choosing the *Selection* button and provide a selection criteria, for example `SOURCESYSTEM = XX01`.
4. Once fully set, activate the *View* function – this function then must be used as input during processing instead of using the Model RDL function directly.

Dynamic Reading Mechanism, Selection using Parameter or Field Model RDL (Join)

SAP Profitability and Performance Management reads data records from a specific result type through active and generated SQL or HANA Views. Automatically the system will consume all data records written in the HANA view.

In order to filter results with dynamic selection, you have to create a SAP Profitability and Performance Management *Join* function on top of the Model RDL.

> ### Example
>
> `SOURCESYSTEM = I_LEGAL_PARAM` where `SOURCESYSTEM` is a field and `I_LEGAL_PARAM` is a parameter or `SOURCESYSTEM = LEGAL_FIELD` where `SOURCESYSTEM` is a field and `LEGAL_FIELD` is a field.

To perform this, follow bellow steps:

1. Create a *Join* function.
2. Select this join function. On the right side of the screen, choose *Edit*.
3. Go to the *Rules* tab and add a rule type “FROM” with the Model RDL as its input.
4. Add a selection by going to the Complex Selection of the rule and use below format to select

```
WHERE SOURCESYSTEM = I_LEGAL_PARAM
```

5. Once fully set, activate the join function – it then must be used as the input during processing instead of using the Model RDL function directly.

Reading while using Parallelization Model RDL (View, Join)

In order to have a more optimal parallelization setup while consuming data from RDL, you can use a SAP Profitability and Performance Management processing function such as *View* or *Join* function on top with a Package Selection entry.

To perform this, follow below steps:

1. Create a processing function such as View with the Model RDL as the input function.
2. On the left side of the screen choose *Edit*.
3. Choose *Function Attributes* button (button on the left of the *Activation* button).
4. On the *Advanced* tab under *Package Selection*, add at least one field that must be selected to avoid reading all data from the table.
5. Once fully set, activate the *View* function – it then must be used as the input during processing instead of using the Model RDL function directly.

Writing Mechanism

SAP Profitability and Performance Management writes data records to a specific RDL result type using a *Writer* function with a configured *Output* function pointing to a Model RDL. This model RDL function must then be configured to point to a specific Result Data Area and Result Type.

> **i** Note
>
> When a Model RDL is used as the output function of a Writer, the field *Model Writer Type* is not available in the Writer header as the default writing method is only "Insert".

For additional information refer to Writer [page 323].

Procedure

Function Access

Follow the steps below to access the *Model RDL* function:

1. In the client where SAP Profitability and Performance Management is installed, choose *SAP Menu* > *Profitability and Performance Management* > *Modeling* > *Start My Environments* .
2. The *Environment* screen appears in a separate browser window. Choose an existing environment and continue. Within the environment, you can set up the newly added function.

Function Configuration

Follow the steps below to configure the *Model RDL* function:

1. In edit mode, configure the following required fields in the header. For more information about the header, see the *Key Features* section above.
   - *Connection Type*: Choose either “Physical” or “Logical”.
   - *Result Data Area*
   - *Result Type*
   - *View Type*: Choose either “Application” or “Integration”.
2. Choose *Synchronize* on the *Fields* tab.
3. The *Sync Model Fields* dialog appears showing the fields to be imported.
4. Choose the fields to be added, then choose *OK*.
5. Choose *Save* and then choose *Activate*.

Related Information

- For more information about common aspects of SAP Profitability and Performance Management functions, see Concepts for Key Users [page 6].
- For more information on how to add and remove functions in SAP Profitability and Performance Management environments, see Function Hierarchy [page 63].

1.4.1.5 Model Table

Model Table is a data model function that allows you to define and access local and remote database tables.

Key Features

Header

In the header, you can choose any of the following connection types:

- If you select the connection type “Logical”, only the field *Connection Name* is available for selection.
- If you select the default connection type “Physical” the following Model Table sources are available for selection:
  - *Environment*: The data model is managed in the Environment. You can use the *Transport Data* option to decide if the data in the model table is transported together with the environment through the system landscape or not.

> **i** Note
>
> Model Table source “Environment” does not have any lifecycle management because it is offered as a temporary data store. It is not intended to fulfill traceability and auditability requirements.

> For those purposes, other data stores are available, like Business Warehouse InfoCubes/ADSOs or SAP HANA History tables.

- *Data Dictionary*: The data model is managed externally and it is referenced within the *Model Table* function to make it available in the environment. For this, you need to enter the table name. Field information is synchronized into the environment fields.
- *HANA*: The data model is managed externally and it is referenced within the *Model Table* function to make it available in the environment. You need to enter the authoring schema and table name. Field information is synchronized into the environment fields.
- *SDA*: The data model is managed externally in a remote system and it is referenced within the *Model Table* function to make it available in the environment. You need to enter the remote source name, remote database, remote schema and remote table name. Field information is synchronized into the environment fields.

Fields

If the Model Table source is Environment, you can enter the fields of the model table as a list in the *Fields* tab.

If the Model Table source is Data Dictionary, SAP HANA or SDA, the system automatically reads the selected table and makes field proposals in the *Sync Model Fields* dialog when you choose *Synchronize*. The field state denotes whether the fields are already available or new fields are added to the environment. You have the option to change the field name and description of the new fields which should be unique. You can also exclude certain fields from read access. If you want the changes to be reset to the initial state of the field, choose *Reset Proposal*.

Further Details

If a DDIC table that is used as the source for a model table has a client field (field with DDIC data type `CLNT`), the system selects data differently, depending on whether or not you have selected the *Exclude* option. If you do not select *Exclude*, the system filters source data, and selects only data for the current system client. If you select *Exclude*, the system selects all data from the source object, and does not filter by client.

If a model table that uses a DDIC table as a source is used as the target for a writer function, the system always poulates the client field with the value of the current client, irrespective of whether or not you have selected the *Exclude* option for the client field.

Procedure: Using the Environment as Model Table Source

Follow the steps below to create a *Model Table* function in the environment:

1. In edit mode, choose the *Add* button.
   The *Function* window appears.
2. Fill in the required fields:
   - *Function*
   - *Description*
   - *Function Type* = “Model Table”

3. Fill in optional fields as necessary:
    - *Event Handling*
    - *Logging*
    - *Processing Type*
    - *Partitioning*
4. Choose *Ok* and save your changes.
5. In edit mode, fill in the following function details:
    - *Connection Type*: Choose either “Physical” or “Logical”.
    - *Model Table Source* = “Environment”
    - *Transport Data*
6. On the *Fields* tab, add the fields needed for the table:
    1. Choose *Add*.
       The *Add Fields* window appears.
    2. Select all the fields needed for the table from the list.
    3. Choose *Ok*.
    4. The fields are now available in the *Fields* tab.
7. Activate the function.
8. In edit mode on function details, choose *Maintain Data*
   The *Data Editor* appears.
   This allows you to load the data in the model table. To maintain the data follow the steps below:
    1. Choose *Edit*.
    2. Input all the data and save.

> **i Note**
>
> The data can be maintained in two ways:
> - Writing directly on the fields (Copy and Paste)
> - Importing using an excel file.

Related Information

For more information about common aspects of SAP Profitability and Performance Management functions, see Concepts for Key Users [page 6].

1.4.1.6 Model View

Model View is a data model function that allows read access to local and remote database tables and views.

Key Features

Header

In the header, you can choose any of the following connection types:

- If you select the connection type “Logical”, only the field *Connection Name* is available for selection.
- If you select the default connection type “Physical” the following Model View sources are available for selection:
  - *Data Dictionary Table*: The table is managed externally and is referenced within the *Model View* function to make it available in the environment. You need to enter the table name. Field information is synchronized into the environment fields.
  - *Data Dictionary View*: The view is managed externally and is referenced within the *Model View* function to make it available in the environment. You need to enter the view name. Field information is synchronized into the environment fields.
  - *HANA Table*: The table is managed externally and it is referenced within the *Model View* function to make it available in the environment. You need to enter the authoring schema and table name. Field information is synchronized into the environment fields.
  - *HANA View*: The view is managed externally and is referenced within the *Model View* function to make it available in the environment.You need to enter the authoring schema and view name. Field information is synchronized into the environment fields.
  - *SDA*: The table or view is managed externally in a remote system and is referenced within the *Model View* function to make it available in the environment. You need to enter the remote source name, remote database, remote schema and remote table name. Field information is synchronized into the environment fields.
  - *CDS View*: The view is defined for existing database tables and any other views or CDS views in the ABAP Dictionary using the statement `DEFINE VIEW` in the CDS DDL in ABAP Core Data Services (CDS). This is done in the CDS source code of a CDS data definition in the ABAP Development Tools.

Fields

The fields of the table or view are listed on the *Fields* tab. You can also exclude fields from read access.

Parameters

If the referenced table or view has parameters, they are listed on this tab. For each parameter, you need to assign either a constant value or an environment parameter.

Further Details

If a DDIC table overview that is used as source for a model view has a client field (field with DDIC data type `CLNT`), the system selects data differently, depending on whether or not you have selected the *Exclude* option.

If you do not select the *Exclude* option, the system filters source data, and selects only data for the current system client. If you select the *Exclude* option, the system selects all data from the source object, and does not filter by client.

Procedure: Using the Data Dictionary Table as Model View Source

Follow the steps below to create a *Model View* function in the environment:

1. In edit mode, choose the *Add* button.
   The *Function* window will appear.
2. Fill in the required fields:
   - *Function*
   - *Description*
   - *Function Type* = "Model View"
3. Fill in any optional fields as necessary:
   - *Event Handling*
   - *Processing Type*
   - *Partitioning*
4. Choose *Ok* to save your changes.
5. In edit mode, fill in the following function details:
   - *Model View Source* = "Data Dictionary Table"
   - *Connection Type*: Choose either "Physical" or "Logical".
   - *Table Name*
6. On the *Fields* tab, choose *Synchronize*.
   The fields of the Data Dictionary (DDIC) table are listed.
7. Activate the function.

Related Information

For more information about common aspects of SAP Profitability and Performance Management functions, see Concepts for Key Users [page 6].

1.4.2 Processing Functions

These functions process data from information functions and produce an output. Processing functions can be connected so that the output of a function is used as input for subsequent functions.

Most functions belong to this category, such as the following:

- Allocation [page 139]
- Calculation [page 150]

- Condition [page 178]
- Conversion [page 181]
- Derivation [page 187]
- Flow Modeling [page 191]
- Funds Transfer Pricing [page 253]
- Join [page 267]
- Transfer Structure [page 282]
- Valuation [page 289]
- View [page 294]

1.4.2.1 Allocation

The *Allocation* function is used to distribute key figures from one entity to another using a distribution base.

The entity from which key figures are distributed is known as the sender. The sender key figures represent the values to be allocated by the allocation function.

The entity that receives the distributed key figures is known as the receiver. One or more key figures from the receiver constitute the distribution base or bases.

The following table explains the key features available:

Key Features

Header

In the header, you define the principal behavior of the allocation.

- Allocation Type
  In allocation, you can use the following allocation types according to your needs:
  1. Allocation
     Key figures of the sender entity are distributed to a receiver entity. and these distribution records are the result of the allocation. The key figures of the sender entity are not affected. This type of allocation is typically used in top-down distribution allocations, activity-based costing allocations and other allocations where iterations or postings of allocation results are not necessary.
  2. Allocation with Offset Records
     Key figures of the sender entity are distributed to a receiver entity, and these distribution records are the result of the allocation. In addition, offset records are created at the granularity of the sender entity but with opposite signs of the key figures. This type of allocation is typically used in assessment allocations and can be iterative, where the sender needs to be reduced by the allocated key figures and the receiver is enhanced, so that the overall sum of the key figure values stays the same.
  3. Allocation with Detailed Offset Records
     This allocation type is similar to “Allocation with Offset Records”. However, here the offset records are created using the sender and receiver entity dimensions to produce more details. This means that information about which fraction of the sender entity key figures were distributed to which receiver record is stored in the results. This type of allocation is typically used in distribution allocation to provide traceability from sender to receiver. It can also be used for iterations.

- Iterative
  If you set the *Iterative* indicator to "Yes", the allocation process is executed iteratively. This action also enables the *Iterative Processing* function on the *Advance* tab of allocation.
  If the modeler selects iterative processing, the system repeats the allocation process, using allocation results from the previous iteration as the sender for the next iteration. Iterations are repeated until there are no senders to be allocated or the exit condition defined in the advanced allocation settings is fulfilled. The modeler defines the exit condition using the early exit check and/or the cycle maximum value. Iteration is repeated until one of these conditions are fulfilled.
- Periodic
  The modeler uses the *Periodic* indicator to specify whether the allocation process is executed based on a defined period or time interval.
  If the modeler selects periodic processing, the system runs the allocation process using the period/time interval defined in the allocation settings in the *Advanced* tab of allocation.
  On the *Advanced* tab, the modeler can specify the fiscal year and period intervals. The modeler can also choose whether periodic processing is cumulative.
  If the *Periodic* option is not meant to be used, the modeler must set it to "No".
- Value Adjustment

During the allocation process, there can be a difference in the sender amount and total receiver amount (to which the given sender amount was distributed) as a result of rounding behavior. These differences in value can be compensated using one of the following financial value adjustment options:

1. No Adjustment
   The system does not adjust the difference in value. This is typically used in planning-only scenarios with high values, where "a missing cent" is not relevant.
2. Last Row
   The system adds the value difference to the last receiver (corresponding to the specified sender).
3. Biggest Value Row
   The difference in value is added to the receiver (corresponding to the specified sender) with the highest allocated amount.
4. Absolute Biggest Value Row
   The difference in value is added to the receiver (corresponding to the specified sender) with the highest allocated amount, not taking the +/- sign into account.

All other header options such as *Include original Input Data*, *Result Handling*, *Suppress initial Results* and *Result Model Table* are called "Functional Building Blocks" and are not specific to allocation. For more information about these options, see Header [page 10].

Sender and Receiver

On the *Sender* and *Receiver* tab, you define the input for the allocation function with an option to enrich the data before the allocation process. Typically, the sender points to G/L data and the receiver points to driver data.

Rules

Each allocation rule line defines one segment of an allocation. For each allocation rule, the sender and receiver rules need to be specified.

To do this, proceed as follows:

1. On the *Rule* tab of your allocation function, choose *Add*.
2. An *Add Detail* popup window appears. Enter the following information:

1. Add Level
   If you choose the “Same Level” option , the rule is processed immediately.
   If you choose the “One Level Below” option, the allocation result from the parent rule is used as the sender by the child rule(s). The final allocation result contains the allocation result derived from the lowest-level child rules.

   > **i Note**
   >
   > You can define allocation rules hierarchically by choosing the “One Level Below” option.

2. Rule
   This is a unique technical ID that must be provided by the modeler. Do not use blank spaces. For example, `ALLOC_COST` is a good rule ID whereas `ALLOC COST` is not. Once you have created the rule, you can no longer edit the entry.
3. Description
   Add a description to enable you to distinguish between rules. You can edit the description later if required.

3. Choose a Rule Type:
   1. Direct Rule Type
      Allocates records from sender to receiver with the same characteristics.
   2. Indirect Rule Type
      Allocates records from sender to receiver according to distribution criteria.
   3. Indirect Detailed Rule Type
      Allocates records from sender to receiver according to distribution criteria comprehensively.
4. Adjust the *Sender Rule* tab:
   1. Sender Rule
      Define the values or amounts to be allocated based on the setting configured. The default option, *Posted Amount*, uses the sender input or the result of an allocation rule at a lower level.
   2. Sender Share
      Defines the percentage of the value that is allocated from the sender. Usually this is 100% so that the full value from the sender is allocated.
   3. Sender Value Fields:
      Defines the value fields that have to be allocated from the sender to the receiver.
   4. Mapping Method:
      When senders are allocated to receivers in direct allocation, receivers are matched based on one of the following mapping methods:
      - Empty as value:
        Empty characteristics from the sender are matched only to empty characteristics of the receiver.
      - Empty as any value:
        Empty characteristics from the sender are matched to any value in the characteristics of the receiver. Characteristic values from the receiver are ignored if the characteristics are empty in the sender.

      In other words, the receiver characteristic value is not taken into consideration when the system allocates amounts from the sender if the sender's characteristics are empty.
   5. Subview:
      You can apply further selections, formulas and groupings here if needed.
5. Adjust the *Receiver Rule* tab:
   1. Receiver Rule:

Define how the values from the sender are to be allocated or distributed by choosing one of the following settings:

- Variable Portions
  The receiver input is used and the distribution base acts as a driver.
- Variable Percentages
  The receiver input is used and the distribution base acts as an allocation percentage.
- Variable Factors
  The receiver input is used and the distribution base acts as an allocation factor.
- Variable Even
  The receiver input is used and no distribution base is needed because the sender is evenly distributed.

2. Scale:
   - No scaling:
     The distribution base is not scaled before it is applied.
   - Standard scaling:
     - If the sum of receiver tracing factors is positive or zero, the smallest negative tracing factor is set to zero.
       The other tracing factors are increased accordingly.
     - If the sum of the receiver tracing factors is negative, the smallest positive tracing factor is set to zero.
       The other tracing factors are decreased correspondingly.
   - Absolute value:
     For negative receiver tracing factors the (–) sign is reversed. All the receiver tracing factors are therefore positive.
   - Negative tracing factors to zero:
     Negative tracing factors are set to zero.
   - Smallest negative tracing factor to zero:
     The smallest negative tracing factor is set to zero. All other tracing factors are increased correspondingly.
   - Smallest negative tracing factor to zero, but zero = zero:
     The smallest negative tracing factor is set to zero. All other tracing factors are increased correspondingly. Receivers that used tracing factor "0" before scaling retain the zero.
3. Distribution Base
   Is the basis for the allocation. The specific treatment is dependent on the receiver rule (see above).
4. Driver Result
   If you make an entry here, the allocation calculates the percentage portion based on the distribution base value and retains the driver percentage in the allocation result. These percentage portions are often easier to read by business users than distribution bases, which sometimes have quite small or large values.
5. Subview:
   You can apply further selections, formulas and groupings if needed.

Further options include the scaling of driver values, the distribution base definition as a field or formula and the option of assigning a driver result field to retain the driver percentage in the allocation result.

Advanced Tab

The following settings are relevant for periodic processing:

1. Iterative Processing
   If iterative allocation is set to "Yes" in the header, you need to make additional settings on the *Advanced* tab.
   The following settings are relevant for iterations:
   - Iteration Counter
     If you register a field in this optional setting, the result shows which records have been allocated in which iteration cycle. As a prerequisite, the field has been defined as an action field on the *Signature* tab of the allocation.
   - Cycle Maximum Value
     The number of iterations is limited to the value entered (for example, 100). You can also use a formula, field or parameter to have a more flexible value set.
   - Early Exit Check
     If you register a check in this optional setting, it determines the processing of the next iteration that fulfills the check condition. This is helpful if you want to apply a threshold. For example, if you define the check "Amount is greater than or equal to 100 (Amount>=100)", it means the allocation result for the next iterations allocates amounts only if they are greater than or equal to 100, and stops further allocation for amounts less than 100.
2. Periodic Processing
   If periodic allocation is set to "Yes" in the header, you need to make additional settings on the *Advanced* tab. The following settings are relevant for periodic processing:
   - Periodic Counter
     This field in the output provides information about the period for which a particular allocation record is created in periodic processing.
     As a prerequisite, the field has been defined as an action field in the *Signature* tab of the allocation.
   - Fiscal Year
     This field contains financial year information in the sender and/or receiver data.
     As a prerequisite, the field has been defined as a selection field in the *Signature* tab of the allocation.
   - Fiscal Year Value
     This is the financial year for which the allocation is executed. The value you enter here is used to restrict sender and receiver data for a given financial year. You can also use a formula, field or parameter to have a more flexible value set.
   - Period
     This field contains term or timeframe information in the sender and/or receiver data.
     As a prerequisite, the field has been defined as a selection field in the signature of the allocation.
   - First Period Value
     Period signifying the start of the analysis or processing timeframe.
     The *First* period field contains the first period for which the allocation can be executed. You can also use a formula, field or parameter to have a more flexible value set.
   - Last Period Value
     Period signifying the end of the analysis or processing timeframe.
     The *Last* period field contains the last period for which allocation can be performed.
   - Specific Periodic Processing
     When you carry out periodic allocation, you can choose the following options for special processing:
     - None

In periodic processing, senders from a given period are allocated to receivers from the same period.

- Cumulation Indicator
  You use this to specify whether the cumulation effect applies to the periodic allocation process. If this indicator is set, the application allocates the sender amounts to receivers posted up to and including the current period. This is based on allocated tracing factors accumulated from the first period onward. The application also accumulates the allocation amounts it has determined and posts them in the current period, minus the amounts allocated in the prior periods.

3. Offset Mapping
   Offset mapping allows you to define characteristics or settings for offset records generated during the allocation process.
   If allocation is relevant for offset data (for example, the allocation type is “Allocation with Offset Records” or “Allocation with detailed Offset Records”), offset data is added to the allocation result. Offset mapping is also used in iterative allocation to define the field mapping to determine sender data from the allocation result of the previous iteration. For example, the receiver cost center in the allocation result of the previous iteration is mapped as the sender cost center in the sender data for the next iteration.
   You can specify the following types of offset mapping:
   1. Offset:
      Field mapping. In other words, the field and relevant offset field is defined.
   2. Debit/Credit
      The field that contains debit/credit information is selected. Values for the debit and credit sign are also set.

Procedure

Function Access

Follow the steps below to access the *Allocation* function:

1. In the client where SAP Profitability and Performance Management is installed, choose *SAP Menu* > *Profitability and Performance Management* > *Modeling* > *Start My Environments* .
2. The *Environment* screen appears in a separate browser window. Choose an existing environment and continue. Within the environment, you can set up the newly added function.

Function Configuration

Follow the steps below to configure the *Allocation* function:

1. In edit mode, configure the following required fields in the header. For more information about the header, see the *Key Features* section above.
   - *Allocation Type*
   - *Include Original Input Data*
   - *Result Handling*
   - *Suppress Initial Result*
   - *Iterative*
   - *Periodic*
   - *Value Adjustment*

2. Define the input function to be used on the *Sender* tab. Afterwards, define the input function to be used on the *Receiver* tab.

   > **i Note**
   >
   > The procedures for the *Sender* tab and the *Receiver* tab are the same as for the *Input* tab. For more information, see Input [page 11].

3. Define the fields to be used on the *Signature* tab.

   > **i Note**
   >
   > For more information about the *Signature* tab, see Signature [page 13].

4. On the *Rules* tab, each allocation rule line defines one segment of an allocation. For more information about the *Rules* tab, see the *Key Features* section above.
5. Define the checks to be used on the *Checks* tab, if needed.

   > **i Note**
   >
   > For more information about the *Checks* tab, see Checks [page 17].

6. Define the required fields on the *Advanced* tab, if needed. For more information about the *Advanced* tab, see the *Key Features* section above.
7. Choose *Save* and then choose *Activate*.

Related Information

- For more information about the iterative feature of the *Allocation* function, see Iterative Feature of the Allocation Function.
- For more information on how to add and remove functions in SAP Profitability and Performance Management environments, see Function Hierarchy [page 63].
- For more information about common aspects of SAP Profitability and Performance Management functions, see Concepts for Key Users [page 6].

1.4.2.1.1 Example: Indirect Detailed Allocation with Offset Records

Sender

Cost Center	Amount
CC01	200
CC02	300

Cost Center	Amount
CC03	400

Receiver

Contract	Product	Distribution Rate
DD01	A100	20
DD05	A100	40
DD03	B200	10
DD04	A100	60
DD02	B200	30

Result

Amount	Cost Center	Contract	Distribution Rate	Product	Portion
25	CC01	DD01	20	A100	0.125
50	CC01	DD05	40	A100	0.25
12.5	CC01	DD03	10	B200	0.062
75	CC01	DD04	60	A100	0.375
37.5	CC01	DD02	30	B200	0.187
37.5	CC02	DD01	20	A100	0.125
75	CC02	DD05	40	A100	0.25
18.75	CC02	DD03	10	B200	0.062
112.5	CC02	DD04	60	A100	0.375
56.25	CC02	DD02	30	B200	0.187
50	CC03	DD01	20	A100	0.125
100	CC03	DD05	40	A100	0.25
25	CC03	DD03	10	B200	0.062
150	CC03	DD04	60	A100	0.375
75	CC03	DD02	30	B200	0.187
–200	CC01		0		0
–300	CC02		0		0
–400	CC03		0		0

1. Amount in sender will be allocated proportionally using *Distribution Rate* in receiver as the distribution percentage.
2. In order to get the portion percentage:
    1. Get the total of *Distribution Rate* from the receiver (*Distribution Rate* = 160).

2. *Portion* is the quotient when you divide *Distribution Rate* by the total of distribution rates (for example, 20 / 160 = 0.1250).
3. Allocate *Amount* to the receiver by multiplying *Amount* with *Portion*. (200 * 0.1250).

3. The negative entries in the *Amount* column are the allocated amounts that came from the sender.

1.4.2.1.2 Example: Direct Allocation with Unassigned Items

Sender

Product	Channel	Customer	Amount	Financial Period
238	92H2	AA	300	1
224	92H2	DD	200	2
238	92H2	AA	400	3
224	92H2	DD	400	4
239	92H3	CC	1,000.00	5

Receiver

Product	Coverage	Channel	Customer	Distribution Rate
224	6981	92H2	DD	60
224	6982	92H2	DD	40
238	6985	CXH0	DD	55
238	6986	CXH0	DD	45
238	6989	92H2	AA	20
238	6990	92H2	AA	80

Result

Product	Channel	Customer	Coverage	Distribution Rate	Amount	Financial Period	Portion
238	92H2	AA	6989	20	60	1	0.2
238	92H2	AA	6990	80	240	1	0.8
224	92H2	DD	6981	60	120	2	0.6
224	92H2	DD	6982	40	80	2	0.4
238	92H2	AA	6989	20	80	3	0.2
238	92H2	AA	6990	80	320	3	0.8
224	92H2	DD	6981	60	240	4	0.6
224	92H2	DD	6982	40	160	4	0.4

Unassigned Item(s)

Product	Channel	Customer	Amount	Financial Period
239	92H3	CC	1,000.00	5

No Matching Characteristics from the Sender

Product	Coverage	Channel	Customer	Distribution Rate
238	CXH0	DD	6985	55
238	CXH0	DD	6986	45

1. The system allocates the amount in sender using the computed percentage based on the *Distribution Rate* of the receiver.
2. To ensure that the correct amount is allocated from the sender to the receiver, proceed as follows:
    1. As an example, use the first record from the sender and locate matching records from the receiver that have the same characteristics (Product = 238, Channel = 92H2 and Customer = AA). In this example, the receivers with characteristics that match those of the sender are the last two with distribution rate 20 and 80.
    2. Calculate the total distribution rate for the grouped items (Product = 238, Channel = AA, Customer = AA) from the last two records in the receiver. In this case, it is 20 and 80, which produces a total distribution rate of 100.

        > **i Note**
        >
        > To calculate the driver result (*Portion*), divide the distribution rates from the receiver (20 and 80) by the total distribution rate (for example 20 / 100 = 0.2).

    3. Multiply the amount provided by the sender (300) with the calculated portion (for example, 300 * 0.2 = 60). This is then the amount allocated to the corresponding record in the receiver.
    4. Follow the same logic for all the remaining sender records (records 2 to 5).
3. After performing steps 1 and 2, four entries from the sender are allocated to the receiver. Since the 5th entry has no direct match with the receiver's data records, it is not allocated, and results in an unassigned item.

> **i Note**
>
> 1. Unassigned Item: In the allocation process, the following error message appears, caused by the 5th record: `Processing Message "Unassigned Items" for Volume=1000 and Quantity=1`.
> 2. The system ignores records in the receiver with no matching characteristics from the sender during the allocation process. Therefore, they are not part of the result after allocation. These records are items with Channel = CXH0.

1.4.2.1.3 Example: Periodic Processing

In this scenario we test a direct allocation with periodic processing, which only shows the processed data (*Include Original Input Data* = "No") with a clear indication to only display the records which satisfy the rule condition.

Sender

Product	Channel	Customer	Amount	Financial Period
238	92H2	AA	300	1
224	92H2	DD	200	2
238	92H2	AA	400	3
224	92H2	DD	400	4
238	92H2	AA	300	5
238	CXH0	DD	1,000	6

Receiver

Product	Coverage	Channel	Customer	Distribution Rate
224	6981	92H2	DD	60
224	6982	92H2	DD	40
238	6985	CXH0	DD	55
238	6986	CXH0	DD	45
238	6989	92H2	AA	20
238	6990	92H2	AA	80

Configuration

Rule	Description	State	Rule Type
DA	Direct Allocation	Active	Direct

Sender Rule	
Sender Rule	Posted Amounts
Sender Share	100
Sender Value Fields	Amount
Mapping Method	Empty as Value

Receiver Rule	
Receiver Rule	Variable Portions
Scale	No Scaling
Distribution Rate	Distribution Rate
Driver Result	Portion

Advanced Tab / Periodic Processing Section	
Periodic Counter	
Fiscal Year	
Total Periods	12

Advanced Tab / Periodic Processing Section

Period	Financial Period
First	1
Last	4
Specific Period Processing	none

Result

Channel	Coverage	Customer	Distribution Rate	Financial Period	Product	Amount	Portion
92H2	6989	AA	20	1	238	60	0.2
92H2	6990	AA	80	1	238	240	0.8
92H2	6982	DD	40	2	224	80	0.4
92H2	6981	DD	60	2	224	120	0.6
92H2	6989	AA	20	3	238	80	0.2
92H2	6990	AA	80	3	238	320	0.8
92H2	6982	DD	40	4	224	160	0.4
92H2	6981	DD	60	4	224	240	0.6

Procedure

1. `ZE_AMT` in the sender is allocated proportionally using `ZE_DRBRT` in the receiver as the distribution percentage (for example, by color).
2. The following steps show hot to get the portion percentage:
    1. Group `ZE_CUST`. `ZE_CHNL` and `ZE_PROD`. These are the fields that have the same characteristics from our sender and receiver.
    2. Get the total of `ZE_DRBRT` of the grouped `ZE_CUST`. `ZE_CHNL` and `ZE_PROD` (`ZE_DRBRT` = 160)
    3. Divide `ZE_DRBRT` by the total of `ZE_DRBRT` (for example 20 / 100 = 0.2)
    4. Allocate `ZE_AMT` to the receiver by multiplying `ZE_AMT` with `ZE_POR` (for example 300 * 0.2 = 60).

Only the first 4 entries of `ZE_AMT` are be allocated to the receiver because periodic processing was specified to end at 4.

1.4.2.2 Calculation

Calculation is a data enrichment function that can be used to enhance the data in a dataset with calculated attributes based on predefined rules at runtime.

The enriched data can then be used for consumption in downstream processes such as allocation. If the data to be calculated is already available in the source data, the calculated data is only overwritten if the condition values are met. Otherwise, the source values are retained.

Key Features

Header

The *Calculation* function includes a parser to detect dependencies between fields used in formulas and to ensure that rules are executed in the correct order internally. Circular dependencies are not allowed.

In the header, you define the principal behavior of the calculation.

You can use the calculation type as a specific header option:

- Absolute [page 151]
- Relative [page 157]
- Workbook Adapter [page 162]

> **i** Note
>
> The Workbook Adapter calls the Calculation Workbook API offered by SAP Profitability and Performance Management Cloud. For it to work, please perform the technical setup to establish a connection between SAP Profitability and Performance Management 3.0 and SAP Profitability and Performance Management Cloud.

The system applies the calculation formula to the grouping result.

Rules

Each calculation rule semantically defines an if-then statement. The “if” part specifies for which records of the input data the rule is relevant. The “then” part is an action and contains a list of fields and formulas that have to be calculated.

Related Information

For more information about common aspects of SAP Profitability and Performance Management functions, see Concepts for Key Users [page 6].

For more information about using the *Calculation* function, see SAP PaPM: Sequence of Calculation Functions Use Case.

For more information on how to set up the *Calculation* function of type “Workbook Adapter”, see Administration Guide for SAP Profitability and Performance Management > *Installation Information* > *Implementation Sequence* > *Calculation Workbook Adapter Setup*.

1.4.2.2.1 Calculation Type: Absolute

“Absolute” means that literally the next value after the value with selection condition is chosen. The selected conditions define a subset of values from the input data.

The selected conditions define a subset of the input data. The function applies the calculations to this subset where the selected conditions are met. This is typically used in planning calculations, where calculations need to be applied to selected line items.

> **i Note**
>
> In the calculation type “Absolute”, the fields *Granularity* and *Selection* are used not only for defining the signature. They also serve as fields for grouping and ordering: The result of the calculation is grouped and ordered based on the fields defined in the granularity and selection.

Procedure

Function Access

Follow the steps below to access the *Calculation* function of type “Absolute”:

1. In the client where SAP Profitability and Performance Management is installed, choose *SAP Menu* > *Profitability and Performance Management* > *Modeling* > *Start My Environments*.
2. The *Environment* screen appears in a separate browser window. Choose an existing environment and continue. Within the environment, you can set up the newly added function.

Function Configuration

Follow the steps below to configure the *Calculation* function of type “Absolute”:

1. In edit mode, configure the following required fields in the header. For more information about the header, see the *Key Features* section in Calculation [page 150].
   - *Calculation Type* = “Absolute”
   - *Result Handling*
   - *Include original Input Data*
   - *Result Model Table*
   - *Suppress initial Result*
2. Define the input function to be used on the *Input* tab.

   > **i Note**
   >
   > For more information about the *Input* tab, see Input [page 11].
3. Define the function to be used on the *Lookup* tab, if needed.

   > **i Note**
   >
   > For more information about the *Lookup* tab, see Lookup [page 12].
4. Define the fields to be used on the *Signature* tab.

   > **i Note**
   >
   > For more information about the *Signature* tab, see Signature [page 13].

5. On the *Rules* tab, each calculation rule defines semantically an if-then statement. The “if”-part specifies the relevant records of the rule's input data. The “then”-part is an action, and contains a list of fields and formulas that have to be calculated.
   Proceed as follows to configure the rules:
   1. On the *Rule* tab of your *Calculation* function, choose + (*Add*).
   2. The *Add Details* screen is displayed. Enter the following information:
      - *Rule*
      - *Description*
   3. Choose *OK*.
   4. Select the created rule.
   5. In the *Selection* section of the created rule, choose + (*Add*).
   6. On the *Fields* screen that is displayed next, choose the field to be used and then choose *OK*.
   7. Select the field and choose *Selection Condition*.
   8. On the *Selection Condition* screen, define the selection condition to be used and choose *OK*.
   9. In the *Action* section of the created rule, choose + (*Add*).
   10. On the *Fields* screen that is displayed next, choose the field to be used and choose *OK*.
   11. Select the field and choose *Formula*.
   12. On the *Formula* screen that is displayed next, define the formula to be used and choose *OK*.
6. Define the checks to be used on the *Checks* tab, if needed.

   > **i** Note
   >
   > For more information about the *Checks* tab, see Checks [page 17].

7. Choose *Save* and then choose *Activate*.

Related Information

- For more information about common aspects of SAP Profitability and Performance Management functions, see Concepts for Key Users [page 6].
- For more information on how to add and remove functions in SAP Profitability and Performance Management environments, see Function Hierarchy [page 63].

1.4.2.2.1.1 Example: Calculation Scenario with Condition

This scenario shows how the “Absolute” calculation type filters out values where the selection conditions are met for both rules and how it calculates the premium based on the formula used.

Input

This is the table that will be used as an input function:

CA - Absolute and Relative Table

Channel	Account	Customer	Product	Quantity	Amount
90AH3	3	CN003	PROD03	15	62.5
90AH04	1	CN001	PROD02	89	250
90AH01	2	CN004	PROD03	99	206.25
90AH04	3	CN006	PROD01	112	132.5
90AH05	3	CN010	PROD03	55	20
90AH01	1	CN001	PROD01	76	125
90AH02	2	CN002	PROD01	80	143.75
90AH02	1	CN004	PROD01	103	175
90AH05	1	CN009	PROD02	126	70.63
90AH04	1	CN001	PROD02	17	100
90AH05	1	CN002	PROD02	13	75
90AH05	1	CN007	PROD01	40	44.75
90AH02	2	CN002	PROD01	20	57.5
90AH03	3	CN005	PROD01	108	153.13
90AH05	1	CN007	PROD01	117	111.88
90AH01	2	CN004	PROD03	15	82.5
90AH03	3	CN005	PROD01	10	61.25
90AH05	1	CN009	PROD02	75	28.25
90AH02	1	CN004	PROD01	13	70
90AH05	1	CN002	PROD02	94	187.5
90AH05	3	CN010	PROD03	130	50
90AH01	1	CN001	PROD01	10	50
90AH04	3	CN006	PROD01	20	53
90AH04	3	CN008	PROD02	50	36.5
90AH03	3	CN003	PROD03	85	156.25

Channel	Account	Customer	Product	Quantity	Amount
90AH04	3	CN008	PROD02	121	91.25
90AH05	1	CN010	PROD03	130	50
90AH01	3	CN001	PROD01	10	50
90AH04	3	CN006	PROD01	20	53
90AH04	3	CN008	PROD02	50	36.5
90AH03	3	CN003	PROD03	85	156.25
90AH04	3	CN008	PROD02	121	91.25

The scenario will filter out values where the selection conditions are met for both rules.

CA - Absolute and Relative Table

Chanel	Account	Customer	Product	Quantity	Amount	Premium
90AH3	3	CN003	PROD03	15	62.5	
90AH4	1	CN001	PROD02	89	250	937.5
90AH1	2	CN004	PROD03	99	206.25	
90AH4	3	CN006	PROD01	112	132.5	1,100
90AH5	3	CN010	PROD03	55	20	
90AH1	1	CN001	PROD01	76	125	11,500
90AH2	2	CN002	PROD01	80	143.75	18,025
90AH2	1	CN004	PROD01	103	175	8,899.38
90AH5	1	CN009	PROD02	126	70.63	18,025
90AH4	1	CN001	PROD02	17	100	8,899.38
90AH5	1	CN002	PROD02	13	75	1,700
90AH5	1	CN007	PROD01	40	44.75	1,150
90AH2	2	CN002	PROD01	20	57.5	16,538.04
90AH3	3	CN005	PROD01	108	153.13	13,089.96
90AH5	1	CN007	PROD01	117	111.88	1,237.5
90AH1	2	CN004	PROD03	15	82.5	
90AH3	3	CN005	PROD01	10	61.25	2,118.75
90AH5	1	CN009	PROD02	75	28.25	612.5
90AH2	1	CN004	PROD01	13	70	17,624
90AH5	1	CN002	PROD02	94	187.5	910
90AH5	3	CN010	PROD03	130	50	
90AH1	1	CN001	PROD01	10	50	1,060
90AH4	3	CN006	PROD01	20	53	1,825
90AH4	3	CN08	PROD02	50	36.5	1,060

Chanel	Account	Customer	Product	Quantity	Amount	Premium
90AH3	3	CN003	PROD03	85	156.25	
90AH4	3	CN008	PROD02	121	91.25	13,281.25
90AH5	3	CN010	PROD03	130	50	
90AH1	1	CN001	PROD01	10	50	1,060
90AH4	3	CN006	PROD01	20	53	1,825
90AH4	3	CN008	PROD02	50	36.5	1,060
90AH3	3	CN003	PROD03	85	156.25	
90AH4	3	CN008	PROD02	121	91.25	13,281.25

Rule 1: We set the condition for *Product* as "PROD01" and use the following formula to calculate the premium: Quantity[1] * Amount[1].

Rule 2: We set the condition for *Product* as "PROD02" and use the following formula to calculate the premium: Quantity[–1] * Amount[–1].

> **i** Note
>
> [1] means that referencing the selection condition of the rule Product = PROD01, the system will calculate the value of the *Premium* of the next absolute data (any succeeding data which satisfies the selection condition).
>
> [-1] means that referencing the selection condition of the rule Product = PROD02, the system will calculate the value of the *Premium* of the previous absolute data (any preceding data which satisfies the selection condition).

Logic and Computation

Example 1

- Rule 1
  Example is PROD01 at Product row 4, the next absolute data is PROD03 at Product row 5. Therefore, value of Premium row 4 = Quantity row 5 * Amount row 5, so 55 * 20 = 1100.
- Rule 2
  Example is PROD02 at Product row 2, the previous absolute data is PROD03 at Product row 1. Therefore the value of Premium is the product of Quantity row 1 * Amount row 1. Hence, the value of Premium at Cell H241 is 937,5.

Example 2

- Rule 1
  Example is PROD01 at Product row 8, the next absolute data is PROD02 at Product row 9. Therefore, value of Premium for row 8 = Quantity of row 9 * Amount of row 9, so 126 * 70,63 = 8899,83.
- Rule 2

Example is PROD02 at Product row 26, the previous absolute data is PROD03 at Product row 25. Therefore, value of Premium for Cell E265 = Quantity of row 25 * Amount of row 25. Hence, value of Premium at Cell H844 = 85 * 156,25 = 13.281,25.

Final Output

Channel	Account	Customer	Product	Quantity	Amount	Premium
90AH4	1	CN001	PROD02	89	250	937.5
90AH4	3	CN006	PROD01	112	132.5	1,100
90AH1	1	CN001	PROD01	76	125	11,500
90AH2	2	CN002	PROD01	80	143.75	18,025
90AH2	1	CN004	PROD01	103	175	8,899.38
90AH5	1	CN009	PROD02	126	70.63	18,025
90AH4	1	CN001	PROD02	17	100	8,899.38
90AH5	1	CN002	PROD02	13	75	1,700
90AH5	1	CN007	PROD01	40	44.75	1,150
90AH2	2	CN002	PROD01	20	57.5	16,538.04
90AH3	3	CN005	PROD01	108	153.13	13,089.96
90AH5	1	CN007	PROD01	117	111.88	1,237.5
90AH3	3	CN005	PROD01	10	61.25	2,118.75
90AH5	1	CN009	PROD02	75	28.25	612,5
90AH2	1	CN004	PROD01	13	70	17,625
90AH5	1	CN002	PROD02	94	187.5	910
90AH1	1	CN001	PROD01	10	50	1,060
90AH4	3	CN006	PROD01	20	53	1,825
90AH4	3	CN008	PROD02	50	36.5	1,060
90AH4	3	CN008	PROD02	121	91.25	13,281.25
90AH1	1	CN001	PROD01	10	50	1,060
90AH4	3	CN006	PROD01	20	53	1,825
90AH4	3	CN008	PROD02	50	36.5	1,060
90AH4	3	CN008	PROD02	121	91.25	13,281.25

1.4.2.2.2 Calculation Type: Relative

“Relative” means that the next value which is equal to the value defined with selection condition is chosen.

The function runs through the complete input data and applies each calculation rule where the selected conditions are met.

> **Note**
>
> This method is similar to the one used for derivations, but respects dependencies in formulas.

Procedure

Function Access

Follow the steps below to access the *Calculation* function of type "Relative":

1. In the client where SAP Profitability and Performance Management is installed, choose *SAP Menu* > *Profitability and Performance Management* > *Modeling* > *Start My Environments*.
2. The *Environment* screen appears in a separate browser window. Choose an existing environment and continue. Within the environment, you can set up the newly added function.

Function Configuration

Follow the steps below to configure the *Calculation* function of type "Relative":

1. In edit mode, configure the following required fields in the header. For more information about the header, see the *Key Features* section in Calculation [page 150].
   - *Calculation Type* = "Relative"
   - *Result Handling*
   - *Include original Input Data*
   - *Result Model Table*
   - *Suppress initial Result*
2. Define the input function to be used on the *Input* tab.

   > **Note**
   >
   > For more information about the *Input* tab, see Input [page 11].
3. Define the function to be used on the *Lookup* tab, if needed.

   > **Note**
   >
   > For more information about the *Lookup* tab, see Lookup [page 12].
4. Define the fields to be used on the *Signature* tab.

   > **Note**
   >
   > For more information about the *Signature* tab, see Signature [page 13].
5. On the *Rules* tab, each calculation rule defines semantically an if-then statement. The "if"-part specifies the relevant records of the rule's input data. The "then"-part is an action, and contains a list of fields and formulas that have to be calculated.
   Proceed as follows to configure the rules:
   1. On the *Rule* tab of your *Calculation* function, choose + (*Add*).
   2. The *Add Details* screen is displayed. Enter the following information:

- *Rule*
- *Description*

3. Choose *OK*.
4. Select the created rule.
5. In the *Selection* section of the created rule, choose + (*Add*).
6. On the *Fields* screen that is displayed next, choose the field to be used and then choose *OK*.
7. Select the field and choose *Selection Condition*.
8. On the *Selection Condition* screen, define the selection condition to be used and choose *OK*.
9. In the *Action* section of the created rule, choose + (*Add*).
10. On the *Fields* screen that is displayed next, choose the field to be used and choose *OK*.
11. Select the field and choose *Formula*.
12. On the *Formula* screen that is displayed next, define the formula to be used and choose *OK*.

6. Define the checks to be used on the *Checks* tab, if needed.

> **i Note**
>
> For more information about the *Checks* tab, see Checks [page 17].

7. Choose *Save* and then choose *Activate*.

Related Information

- For more information about common aspects of SAP Profitability and Performance Management functions, see Concepts for Key Users [page 6].
- For more information on how to add and remove functions in SAP Profitability and Performance Management environments, see Function Hierarchy [page 63].

1.4.2.2.2.1 Example: Calculation with Lookup (Relative)

This scenario shows how the "Relative" calculation type obtains values from another model table using lookup and how it calculates the amount based on the selection condition and formula used.

Input

This is the table that will be used as an input function:

Input 1: CA - Data Table 1

Channel	Account	Customer	Product	Quantity	Amount
90AH5	1	CN007	PROD01	80	17.9
90AH4	3	CN008	PROD02	100	14.6

Channel	Account	Customer	Product	Quantity	Amount
90AH2	2	CN002	PROD01	40	23
90AH5	1	CN002	PROD02	25	30
90AH1	1	CN001	PROD01	20	20
90AH3	3	CN003	PROD03	30	25
90AH3	3	CN005	PROD01	20	24.5
90AH4	1	CN001	PROD02	33	40
90AH2	1	CN004	PROD01	25	28
90AH4	3	CN006	PROD01	40	21.2
90AH1	2	CN004	PROD03	30	33
90AH5	1	CN009	PROD02	75	11.3

This is the table that will be used as the lookup input:

Input 2: CA - Data Table 2

Channel	Account	Customer	Product	Quantity	Amount
90AH3	3	CN003	PROD03	15	62.5
90AH4	1	CN001	PROD02	89	250
90AH1	2	CN004	PROD03	99	206.25
90AH4	3	CN006	PROD01	112	132.5
90AH5	3	CN010	PROD03	55	20
90AH1	1	CN001	PROD01	76	125
90AH2	2	CN002	PROD01	80	143.75
90AH1	1	CN004	PROD01	103	175
90AH2	1	CN009	PROD02	126	70.63
90AH4	1	CN001	PROD02	17	100
90AH5	1	CN002	PROD02	13	75
90AH5	1	CN007	PROD01	40	44.75
90AH5	2	CN002	PROD01	20	57.5
90AH3	3	CN005	PROD01	108	153.13
90AH5	1	CN007	PROD01	117	111.88
90AH1	2	CN004	PROD03	15	82.5
90AH3	3	CN005	PROD01	10	61.25
90AH5	1	CN009	PROD02	75	28.25
90AH2	1	CN004	PROD01	13	70
90AH5	1	CN002	PROD02	94	187.5
90AH5	3	CN010	PROD03	130	50

Channel	Account	Customer	Product	Quantity	Amount
90AH1	1	CN001	PROD01	10	50
90AH4	3	CN006	PROD01	20	53
90AH4	3	CN008	PROD02	50	36.5
90AH3	3	CN003	PROD03	85	156.25
90AH4	3	CN008	PROD02	121	91.25

Calculation

1. Collect all entries containing “90AH5” on the *Channel* field of Data Table 1.

   CA - Data Table 1

Channel	Account	Customer	Product	Quantity	Amount
90AH5	1	CN007	PROD01	80	17.9
90AH5	1	CN002	PROD02	25	30
90AH5	1	CN009	PROD02	75	11.3

2. Collect all entries containing “90AH5” on the *Channel* field of Data Table 2 and sum up the “Amount” field.

   CA - Data Table 2

Channel	Account	Customer	Product	Quantity	Amount
90AH5	3	CN010	PROD03	55	20
90AH5	1	CN009	PROD02	126	70.63
90AH5	1	CN002	PROD02	13	75
90AH5	1	CN007	PROD01	40	44.75
90AH5	1	CN007	PROD01	117	111.88
90AH5	1	CN009	PROD02	75	28.25
90AH5	1	CN002	PROD02	94	187.5
90AH5	3	CN010	PROD03	130	50
				Aggregated Amount of 90AH5	588.01

3. The *Amount* field will be populated with the result of the assigned formula on the *Rules* tab (Quantity * MTCA2.Amount [Channel=90AH5] / 2), where Quantity = Quantity from Data Table 1 and MTCA2./ZQA/ZMT/[Channel=90AH5] = Aggregated Amount of 90AH5 from Data Table 2.

Channel	Account	Customer	Product	Quantity	Amount
90AH5	1	CN007	PROD01	80	23,520.40

   The *Amount* field will then have this formula:
   Quantity * MTCA2.Amount [Channel=90AH5] / 2

When we compute the following values for each field, we will have the following output of amount:
80 * 588,010/2, whereby 588,010 is the aggregated amount of 90AH5 of Data Table 2.

Channel	Account	Customer	Product	Quantity	Amount
90AH5	1	CN002	PROD02	25	7,350.13

The *Amount* field will then have this formula:
Quantity * MTCA2.Amount [Channel=90AH5] / 2
When we compute the following values for each field, we will have the following output of amount:
25 * 588,010/2, whereby 588,010 is the aggregated amount of 90AH5 of Data Table 2.

Channel	Account	Customer	Product	Quantity	Amount
90AH5	1	CN009	PROD02	75	22,050.38

The *Amount* field will then have this formula:
Quantity * MTCA2.Amount [Channel=90AH5] / 2
When we compute the following values for each field, we will have the following output of amount:
25 * 588,010/2, whereby 588,010 is the aggregated amount of 90AH5 of Data Table 2.

Final Output

Channel	Account	Customer	Product	Quantity	Amount
90AH5	1	CN007	PROD01	80	23,520.40
90AH5	1	CN002	PROD02	25	7,350.13
90AH5	1	CN009	PROD02	75	22,050.38

1.4.2.2.3 Calculation Type: Workbook Adapter

The calculation type *Workbook Adapter* allows you to set up calculations in a way similar to the one used for defining formulas and sheets in an MS Excel workbook. Just like rules in other functions, after activation these calculations are executed during the system runtime.

Procedure

Function Access

Follow the steps below to access the *Calculation* function of type “Workbook Adapter”:

1. In the client where SAP Profitability and Performance Management is installed, choose *SAP Menu* > *Profitability and Performance Management* > *Modeling* > *Start My Environments* >.
2. The *Environment* screen appears in a separate browser window. Choose an existing environment and continue. Within the environment, you can set up the newly added function.

Function Configuration

Follow the steps below to configure the *Calculation* function of type “Workbook Adapter”:

1. In edit mode, configure the following required fields in the header:
   - *Calculation Type* = “Workbook Adapter”
   - *Connection Name*: Choose the previously created connection name.
   - *Include Original Input Data*
   - *Result Model Table*

   > **i Note**
   >
   > For more information about the header, see Header [page 10].

2. Define the input function to be used on the *Input* tab.

   > **i Note**
   >
   > For more information about the *Input* tab, see Input [page 11].

3. Define the function to be used on the *Lookup* tab, if needed.

   > **i Note**
   >
   > For more information about the *Lookup* tab, see Lookup [page 12].

4. Define the fields to be used on the *Signature* tab. For more information about the *Signature* tab, see Signature [page 13].

   > **i Note**
   >
   > If you maintain fields on the *Signature* tab, the system displays these fields on the *Input* and *Result* tabs (with table names `_INPUT` and `_RESULT`) in the *Calculation Rules* section. This is particularly useful with the NEO UI when you configure calculation rules. You can add or create dummy data for the field values to test the calculation logic during modeling. These dummy data are not used during the runtime of the system.

5. Define the rules on the *Rules* tab as follows:
   1. Choose *Configure Workbook*. The system redirects you to a new window showing a different user interface (NEO UI) where you can configure the calculation rules.

      > **i Note**
      >
      > The *Calculation Rules* node just contains an empty default sheet (“Sheet1”), even though the configurations have already been made.

   2. To show the *Input* and *Result* tabs (parameters if applicable), follow the steps below:
      1. Choose the *Calculation Rules* node and expand the properties panel on the right-hand side of the screen
   3. Choose (*Maximize*). The system opens the *Workbook* screen showing a tabular view of the workbook.
   4. Choose *Data* from the menu options above the tabular view.
   5. Choose *Synchronize*. Now the *Input* and *Result* tabs appear at the bottom of the workbook's tabular view.

6. Choose (*Exit Fullscreen*) to minimize the *Workbook* screen.
7. The properties panel now displays the *Calculation Rules* > *Workbook* sheet with the *Input* and *Result* tabs that show the fields defined in the *Signature* node.

> **i Note**
>
> If you have configured the *Lookup* node, the system also displays the *<Input Function ID>* tab.

8. Like other spreadsheet applications, the system can use formulas from multiple tabs for the calculation. After the calculation has been completed, the system displays the calculation result.

> **i Note**
>
> During modeling, the tabs do not show the data records from your input and lookup tables. Therefore, we recommend adding dummy data in the Excel tabs to test the Excel formula logic. These data are only used for modeling purposes and are not considered in the calculation during system runtime.

9. Optionally, define the parameters to be used on the *Parameters* tab as follows:
    1. Assign the *Workbook Adapter* function to a process template as an activity.

    > **i Note**
    >
    > If the process template is not used for running a *Workbook Adapter* function, the parameter value is set to initial.

    2. Enter a value for the parameter.
10. Optionally, you can import an existing Excel or CSV file with formulas. In this case, the *Workbook Adapter* function incorporates as much of the calculation logic as possible.

> **i Note**
>
> Provide the data of the imported file in “TABLE” format to ensure that the function processes the data correctly.

Import files as follows:

1. Choose (*Maximize*) to open the *Workbook* screen.
2. Choose the *File* button located at the upper left of the workbook.
3. Choose *Import*.
4. Depending on the type of file to be imported, choose either *Import Excel File* or *Import CSV File*.
5. Select the required import file.

6. Optionally, you can rename the tabs of the tabular view on the *Workbook* screen as follows:
    1. Choose (*Maximize*) to open the *Workbook* screen.
    2. Double-click the tab you want to rename.
    3. Enter the new name.

Related Information

For more information on how to set up the *Calculation* function of type "Workbook Adapter", see Administration Guide for SAP Profitability and Performance Management > *Installation Information* > *Implementation Sequence* > *Calculation Workbook Adapter Setup*.

The following blog posts also provide helpful information about this function:

- Workbook Calculation Adapter Overview
- Workbook Calculation (with Additional Created Field)
- Workbook Calculation (with Parameter)
- Workbook Calculation (with Look Up)
- Workbook Calculation (with Import)
- Workbook Calculation Using an HR Expenses Budgeting Worksheet as Input to Calculation

1.4.2.2.3.1 Example: Workbook Adapter with Additional Field

This scenario shows how the *Workbook Adapter* function type processes the data from the Input Model Table. Also, it allows for an additional field to be included in the result that is calculated through a formula maintained directly in a workbook sheet.

Input

You use this table as an input function:

Input Model Table

Customer	Product	Quantity	Amount
CUST01	PROD01	1	10
CUST02	PROD02	2	20
CUST03	PROD03	3	30
CUST04	PROD04	4	40
CUST05	PROD05	5	50
CUST05	PROD0X	50	500
CUST06	PROD06	60	600

On the *Signature* tab, you can specify the following data:

Signature Tab

Granularity	Selection	Action
Customer	Customer	Total Amount

Granularity	Selection	Action
Product	Product	
	Quantity	
	Amount	

You maintain the above fields to perform the following tasks:

1. Enter the fields *Customer* and *Product* in the *Granularity* column as they contain unique data records. This way, you make sure that the system processes data records from the input data in a grouped manner. In addition, you ensure that the system considers all records from the input function for processing.
2. Make sure that the fields *Customer*, *Product*, *Quantity*, and *Amount* (which can also be seen in the Input Model Table) are included in the result table.
3. Add another field used to create a formula which can be included in the final result.

> **i Note**
>
> Make sure that the field is created in the environment, otherwise the system will display an error.

In the *Calculation Rules* worksheet, there are two tabs (*Input* and *Result*). They contain the fields maintained in the *Signature* section.

Customer	Product	Quantity	Amount	Total Amount
		0	0	0

Both the *Input* and the *Result* tab contains the table above.

> **i Note**
>
> The tabs do not show the data records from the input table. We recommend adding dummy data in the Excel tabs to test the Excel formula logic. These data are only used for modeling purposes and are not considered in the calculation during system runtime.

The table below shows the dummy data for the tabs:

Input Tab

Customer	Product	Quantity	Amount	Total Amount
ARR21	SHOES	143	300	0

> **i Note**
>
> Even though dummy data is maintained on the *Input* tab, the system still captures the data from the Input Model Table.

The formulas shown below are used for the *Result* tab:

Result Tab

Customer	Product	Quantity	Amount	Total Amount
=Input!A2	=Input!B2	=Input!C2	=Input!D2	=C2*D2

By maintaining these formulas, you are cross-referencing the values captured by the first four fields of the *Input* tab. The additional field *Total Amount* contains a formula which multiplies the value of the *Quantity* and *Amount* fields.

Output

Final Output Table

Customer	Product	Quantity	Amount	Total Amount
CUST01	PROD01	1	10	10.00
CUST02	PROD02	2	20	40.00
CUST03	PROD03	3	30	90.00
CUST04	PROD04	4	40	160.00
CUST05	PROD05	5	50	250.00
CUST05	PROD0X	50	500	25,000.00
CUST06	PROD06	60	600	36,000.00

The first four fields (*Customer*, *Product*, *Quantity* and *Amount*) just capture the data from the Input Model table because the formula created refers to the input. The additional field (*Total Amount*) multiplies the value from the *Quantity* and *Amount* fields accordingly.

In this example, only one line of formula was created. In any way, the function considers all records from the input function for processing because the fields have been maintained in the *Granularity* fields from the *Signature* section.

1.4.2.2.3.2 Example: Workbook Adapter with Parameter

This scenario shows how the *Workbook Adapter* function processes the data from the Input Model Table. Also, it allows for an additional field to be included in the result that is calculated through a formula with parameters.

Parameters

Create a parameter based on the information below:

Key Figure Parameter			General			
Field	**Description**	**Scale**	**Data Type**	**Data Length**	**Data Decimals**	**Unit Field**
PKW_DISC	Discount Parameter	1	DEC	18	2	

For more information on how to create environment fields or parameters, see Environment Fields [page 55].

Input

You use this table as an input function:

Input Model Table

Customer	Product	Quantity	Amount
CUST01	PROD01	1	10
CUST02	PROD02	2	20
CUST03	PROD03	3	30
CUST04	PROD04	4	40
CUST05	PROD05	5	50
CUST05	PROD0X	50	500
CUST06	PROD06	60	600

On the *Signature* tab, you can specify the following data:

Signature Tab

Granularity	Selection	Action
Customer	Customer	Total Amount
Product	Product	Total Discounted Amount
	Quantity	
	Amount	

You maintain the above fields to perform the following tasks:

1. Enter the fields *Customer* and *Product* in the *Granularity* column as they contain unique data records. This way, you make sure that the system processes data records from the input data in a grouped manner. In addition, you ensure that the system considers all records from the input function for processing.
2. Make sure that the fields *Customer*, *Product*, *Quantity*, and *Amount* (which can also be seen in the Input Model Table) are included in the result table.
3. Add another field used to create a formula which can be included in the final result.

> **i Note**
>
> Make sure that the field is created in the environment, otherwise the system will display an error.

In the *Calculation Rules* worksheet, there are two tabs (*Input* and *Result*). They contain the fields maintained in the *Signature* section.

Customer	Product	Quantity	Amount	Total Amount
		0	0	0

Both the *Input* and the *Result* tab contains the table above.

> **i Note**
>
> The tabs do not show the data records from the input table. We recommend adding dummy data in the Excel tabs to test the Excel formula logic. These data are only used for modeling purposes and are not considered in the calculation during system runtime.

The *Parameter* tab shows the table below. You can manually enter dummy data as input in the *Parameter* field. This helps you understand the calculation of the formula that is to be created on the *Result* tab.

Parameter Tab

Parameter	Description	Value
PKF_DISC	Discount Parameter	0

The table below shows the dummy data for the tabs:

Input Tab

Customer	Product	Quantity	Amount	Total Amount	Total Discounted Amount
ARR21	SHOES	143	300	0	0

> **i Note**
>
> Even though dummy data is maintained on the *Input* tab, the system still captures the data from the Input Model Table.

The formulas shown below are used for the *Result* tab:

Result Tab

Customer	Product	Quantity	Amount	Total Amount	Total Discounted Amount
=Input!A2	=Input!B2	=Input!C2	=Input!D2	=C2*D2	=E2-(E2*Parameters!C2)

By maintaining these formulas, you are cross-referencing the values captures by the first four fields of the *Input* tab. The additional field *Total Amount* contains a formula which multiplies the value of the *Quantity* and *Amount* fields. The second additional field *Total Discounted Amount* calculates the discounted amount of the total amount based on the set parameter for the discount value.

Process Templates

Since the maintained value of parameter in the workbook calculation rules contains only dummy data, you need to maintain the parameter value by creating a process template and execution activity.

Under *Calculation Unit Details* node, create the process template and execution activity, based on the information below. Make sure that the created parameter is specified in the *Parameters* node under *Calculation Unit*.

Process Template – General

Process	Description	Process Type	Process State
PRO_	Process for Workbook Adapter	Run	Active Template

Parameter	Formula	Value Selection
PKF_DISC	0.75	

Process Activity – General

Activity Type	Activity	Description	Activity Function
Execution Activity	A001	Using Parameters in Workbook Adapter	03: Workbook Adapter

Output

Final Output Table

Customer	Product	Quantity	Amount	Total Amount	Total Discounted Amount
CUST01	PROD01	1	10	10.00	2.50
CUST02	PROD02	2	20	40.00	10.00

Customer	Product	Quantity	Amount	Total Amount	Total Discounted Amount
CUST03	PROD03	3	30	90.00	22.50
CUST04	PROD04	4	40	160.00	40.00
CUST05	PROD05	5	50	250.00	62.50
CUST05	PROD0X	50	500	25,000.00	6,250.00
CUST06	PROD06	60	600	36,000.00	9,000.00

The first four fields (*Customer*, *Product*, *Quantity* and *Amount*) just capture the data from the Input Model Table because the formula created has reference to the input. The additional field (*Total Amount*) multiplies the value from the *Quantity* and *Amount* fields accordingly. For the second additional field (*Total Discounted Amount*), the function calculates the total discounted amount based on the maintained formula.

In this example, only one line of formula has been created. In any way, the function considers all records from the input function for processing because the fields have been maintained in the *Granularity* fields from the *Signature* section.

1.4.2.2.3.3 Example: Workbook Adapter with Lookup

This scenario shows how the *Workbook Adapter* function type processes the data from the Input Model Table and a Lookup Table. Please note that – although more than one Lookup Table is allowed – only one such table is used for this scenario. Also, you can include additional fields in the result that is calculated through a formula maintained directly in a workbook sheet.

Input

You use this table as an input function

Input Model Table

Customer	Product	Quantity	Amount
CUST01	PROD05	15	62.50
CUST02	PROD04	89	250.00
CUST03	PROD03	99	206.25
CUST04	PROD02	112	132.50
CUST05	PROD01	55	20.00

The following table is used as a lookup function:

Lookup Model Table (CAU06)

Product	Discount Percentage
PROD30	0.01
PROD29	0.02
PROD28	0.03
PROD27	0.04
PROD26	0.05
PROD25	0.06
PROD24	0.07
PROD23	0.08
PROD22	0.09
PROD21	0.1
PROD20	0.11
PROD19	0.12
PROD18	0.13
PROD17	0.14
PROD16	0.15
PROD15	0.16
PROD14	0.17
PROD13	0.18
PROD12	0.19
PROD11	0.2
PROD10	0.21
PROD09	0.22
PROD08	0.23
PROD07	0.24
PROD06	0.25
PROD05	0.26
PROD04	0.27
PROD03	0.28
PROD02	0.29
PROD01	0.3

On the *Signature* tab, you can enter the following data:

Signature Tab

Granularity	Selection	Action
Customer	Customer	Total Discount
Product	Product	
	Quantity	
	Amount	

You maintain the above fields to perform the following tasks:

1. Enter the fields *Customer* and *Product* in the *Granularity* column as they contain unique data records. This way, you make sure that the system processes data records from the input data in a grouped manner. In addition, you ensure that the system considers all records from the input function for processing.
2. Make sure that the fields *Customer*, *Product*, *Quantity*, and *Amount* (which can also be seen in the Input Model Table) are included in the result table.
3. Add another field used to create a formula which can be included in the final result.

> **i Note**
>
> Make sure that the field is created in the environment, otherwise the system will display an error.

In the *Calculation Rules* worksheet, there are the *Input* and *Result* tabs as well as the *Lookup* tab that relates to the Lookup Table being connected which is named after its function ID (“CAU06”).

The *Input* and *Result* tabs contain the fields maintained in the *Signature* section. The *Lookup* tab contains the fields maintained in the Lookup Tables.

Input and Result Tabs

Customer	Product	Quantity	Amount	Total Discount
		0	0	0

CAU06 (Lookup) Tab

Product	Discount Percentage
	0

> **i Note**
>
> The tabs do not show the data records from the input table. We recommend adding dummy data in the Excel tabs to test the Excel formula logic. These data are only used for modeling purposes and are not considered in the calculation during system runtime.

The table below shows the dummy data for the tabs:

Input Tab

Customer	Product	Quantity	Amount	Total Discount
ARR21	SHOES	143	300	0

CAU06

Product	Discount Percentage
PIPES	90

> **i Note**
>
> Even though dummy data is maintained on the *Input* tab, the system still captures the data from the Input Model Table.

The formulas shown below are used for the *Result* tab:

Result Tab

Customer	Product	Quantity	Amount	Total Discount
=Input!A2	=Input!B2	=Input!C2	=Input!D2	=(C2*D2)*VLOOKUP(B2,CAU06!A:B,2,0)

By maintaining these formulas, you are cross-referencing the values captured by the first four fields of the *Input* tab. The additional field *Total Discount* contains a formula which does the following:

1. Multiply the value of the *Quantity* and *Amount* fields.
2. Lookup the discount percentage from the Look Up table CAU06 according to the value of the field *Product*.
3. Multiply the result of the first two steps.

Output

Final Output

Customer	Product	Quantity	Amount	Total Discount
CUST01	PROD05	15	62.50	243.75
CUST02	PROD04	89	250.00	6,007.50
CUST03	PROD03	99	206.25	5,717.25
CUST04	PROD02	112	132.50	4,303.60
CUST05	PROD01	55	20.00	330.00

The first four fields (*Customer*, *Product*, *Quantity*, and *Amount*) just capture the data from the Input Model Table because the formula created has reference to the input. The additional field (*Total Discount*) multiplies the product of *Quantity* and *Amount* with the discount percentage looked up from the table CAU06.

In this example, only one line of formula has been created. In any way, the function considers all records from the input function for processing because the fields have been maintained in the *Granularity* fields from the *Signature* section.

1.4.2.2.3.4 Example: Workbook Adapter with Import

This scenario shows how the *Workbook Adapter* function type processes the data from the Input Model table and an imported local Excel file. Also, it allows for additional fields to be included in the result that is calculated through a formula maintained directly in a workbook sheet.

Input

You use this table as an input function:

Input Model Table

Customer	Product	Quantity	Amount
CUST01	PROD05	15	62.50
CUST02	PROD04	89	250.00
CUST03	PROD03	99	206.25
CUST04	PROD02	112	132.50
CUST05	PROD01	55	20.00

On the *Signature* tab, you can enter the following data:

Signature Tab

Granularity	Selection	Action
Customer	Customer	Total Discount
Product	Product	
	Quantity	
	Amount	

You maintain the above fields to perform the following tasks:

1. Enter the fields *Customer* and *Product* in the *Granularity* column as they contain unique data records. This way, you make sure that the system processes data records from the input data in a grouped manner. In addition, you ensure that the system considers all records from the input function for processing.
2. Make sure that the fields *Customer*, *Product*, *Quantity*, and *Amount* (which can also be seen in the Input Model Table) are included in the result table.
3. Add another field used to create a formula which can be included in the final result.

> **i Note**
>
> Make sure that the field is created in the environment, otherwise the system will display an error.

In the *Calculation Rules* worksheet, there are two tabs (*Input* and *Result*). They contain the fields maintained in the *Signature* section.

Perform Import

Follow the steps below to import an Excel or CSV file:

1. On the expanded *Workbook* screen, choose the *File* button located at the upper left of the workbook.
2. The system displays a new screen. Choose *Import*.
3. Depending on the type of the file to be imported, choose either *Excel File* or *CSV File* in the *Import* section of the screen. Then choose *Import Excel File* or *Import CSV File* (depending on the selected file type).
4. Select the required import file.

In this example, the data is imported from the following locally saved Excel file:

Discount Percentage

Product	Discount Percentage
PROD30	0.01
PROD29	0.02
PROD28	0.03
PROD27	0.04
PROD26	0.05
PROD25	0.06
PROD24	0.07
PROD23	0.08
PROD22	0.09
PROD21	0.1
PROD20	0.11
PROD19	0.12
PROD18	0.13
PROD17	0.14
PROD16	0.15
PROD15	0.16
PROD14	0.17
PROD13	0.18
PROD12	0.19
PROD11	0.2
PROD10	0.21
PROD09	0.22
PROD08	0.23
PROD07	0.24
PROD06	0.25
PROD05	0.26

Product	Discount Percentage
PROD04	0.27
PROD03	0.28
PROD02	0.29
PROD01	0.3

After the import, the calculation rules are filled with the data from the imported Excel file. The *Input* and *Result* tabs are still intact.

Input and Result Tabs

Customer	Product	Quantity	Amount	Total Discount
		0	0	0

> **i Note**
>
> The tabs do not show the data records from the input table. We recommend adding dummy data in the Excel tabs to test the Excel formula logic. These data are only used for modeling purposes and are not considered in the calculation during system runtime.

The table below shows the dummy data for the tabs:

Input Tab

Customer	Product	Quantity	Amount	Total Discount
ARR21	SHOES	143	300	0

> **i Note**
>
> Even though dummy data is maintained on the *Input* tab, the system still captures the data from the Input Model Table.

The formulas shown below are used for the *Result* tab:

Result Tab

Customer	Product	Quantity	Amount	Total Discount
=Input!A2	=Input!B2	=Input!C2	=Input!D2	=(C2*D2)*VLOOKUP(B2,'Discount Percentage'!A:B,2,0)

By maintaining these formulas, you are cross-referencing the values captured by the first four fields of the *Input* tab. The additional field *Total Discount* contains a formula which does the following:

1. Multiply the value of the *Quantity* and *Amount* fields.
2. Look up the value from the *Discount Percentage* table (the data from the imported file) according to the data in the field *Product*.
3. Multiply the result of the first two steps.

Final Output

Final Output

Customer	Product	Quantity	Amount	Total Discount
CUST01	PROD05	15	62.50	243.75
CUST02	PROD04	89	250.00	6,007.50
CUST03	PROD03	99	206.25	5,717.25
CUST04	PROD02	112	132.50	4,303.60
CUST05	PROD01	55	20.00	330.00

The first four fields (*Customer*, *Product*, *Quantity*, and *Amount*) just capture the data from the Input Model Table because the created formula has reference to the input. The additional field (*Total Discount*) multiplies the product of *Quantity* and *Amount* with the value looked up from the *Discount Percentage* table (the data from the imported file).

In this example, only one line of formula has been created. In any way, the function considers all records from the input function for processing because the fields have been maintained in the *Granularity* fields from the *Signature* section.

1.4.2.3 Condition

The *Condition* function consists only of a *Condition* field that you can maintain using the formula editor. Conditions can be specified for values of defined parameters.

Functions can be placed under the *Condition* function (in addition to the *Calculation Unit* and *Description* function).

The *Condition* function acts as a trigger for the processing logic of its child functions as follows:

- If the specified condition is met, the whole function tree is executed accordingly.
- If the condition is not fulfilled, the system skips the function logic of the child functions and displays a corresponding message which is added to the application log.

Key Features

Rules

You can enter the condition or formula for the values of defined parameters.

Procedure

Function Access

Follow the steps below to access the *Calculation* function of type “Absolute”:

1. In the client where SAP Profitability and Performance Management is installed, choose *SAP Menu* > *Profitability and Performance Management* > *Modeling* > *Start My Environments*.
2. The *Environment* screen appears in a separate browser window. Choose an existing environment and continue. Within the environment, you can set up the newly added function.

Function Configuration

Follow the steps below to configure the *Condition* function:

1. Choose + (*Add*) to open the *Add Function* window.
2. Add a level for the function and fill in the required fields on the *General* tab:
   - *Function*: Specify a name for the function.
   - *Description*: Enter a text to describe the new function.
   - *Function Type*: Select “Condition”.
3. Fill in any optional fields as necessary:
   - *Event Handling*
   - *Processing Type*
   - *Partitioning*
4. Choose *OK* to save your changes.
5. In edit mode, choose the *Formula* button. The *Formula* screen appears.
6. Define parameters or enter a formula.
7. Choose *OK* and then choose *Save* to save your changes.

Related Information

- For more information about common aspects of SAP Profitability and Performance Management functions, see Concepts for Key Users [page 6].
- For more information on how to add and remove functions in SAP Profitability and Performance Management environments, see Function Hierarchy [page 63].

1.4.2.3.1 Example: Single Function Under a Condition

Using a *View* function, this scenario shows a basic way to use the *Condition* function, and how the data and logs behave when the defined parameter condition is fulfilled.

In this scenario, the *Condition* function uses a formula with a parameter value of 10 (for example, `I_KF=10`).

Under the node of the *Condition* function, a Model Table is used as input for the *View* function. The *View* function multiplies the amount by the parameter.

Input Data

Customer	Product	Amount
CUST1	PROD01	100
CUST2	PROD02	200
CUST3	PROD03	300
CUST4	PROD04	400

When you run the *View* function where the package parameter was set to `I_KF`, it can either meet or not meet the condition defined in your *Condition* function.

If the condition `I_KF=10` is met, the amount is multiplied by 10 as shown below:

Customer	Product	Amount
CUST1	PROD01	100 * 10
CUST2	PROD02	200 * 10
CUST3	PROD03	300 * 10
CUST4	PROD04	400 * 10

Result

Customer	Product	Amount
CUST1	PROD01	1000
CUST2	PROD02	2000
CUST3	PROD03	3000
CUST4	PROD04	4000

Log Sample

The logs show that no steps were skipped during the run:

Output Code

```
Run Parameters Packages Package Parameter=l_KF=10. Package Selection=
Input xxxx selected 4 records
Processing Message "OK" for Volume=0.0 and Quantity=4
```

Note

If the condition `I_KF=10` is not met (for example, `I_KF=20`), there is no result.

1.4.2.4 Conversion

The *Conversion* function covers two main types of conversion:

- Currency Conversion [page 182]: Converts one currency into another country's currency.
- Unit Conversion [page 184]: Converts between different units of measurement for the same quantity based on the conversion factors.

Key Features

Headers

All header options, for example *Include Original Input Data*, *Result Handling*, *Suppress Initial Results*, and *Result Model Table* are functional building blocks and are not specific to conversion. For more information about these options, see Header [page 10].

Rules

You can define the type of conversion (currency conversion or unit conversion) by defining the line type within the rule.

Conversion Type Definitions

You need to define which references the system will use to run the conversion function for currency and unit conversions.

For example, category (unit or currency), tables (T006 or TCURR...), from which client, schema, conversion methods, rates, and market relevant data.

1. In the *Environment* section, choose *Conversion Types*.
2. Choose *Edit* in the upper right corner of the *Environment* section.
3. Choose *Add*, make entries in all the the required fields and choose *OK*.

Fields	Description
Conversion Types	This is a unique ID that must be provided by the modeler, for example “CONV0001”.
Description	To distinguish between rules, enter a description. Once the description has been created, editing is still possible.
Category	Specify the conversion category (unit and currency).

The function uses SAP S/4HANA Conversion, for more information about the other fields available in the Conversion definition, see:

- CONVERT_CURRENCY Function
- CONVERT_UNIT Function

1.4.2.4.1 Currency Conversion

The *Currency Conversion* function can convert one currency into another country's currency. Based on current exchange rates, the value of the source amount may increase or decrease after the currency has been converted.

Input Fields:

- *Conversion Type*: Defines the conversion rate, for example the middle rate, or the bid & ask rate for currency conversion
- *Value*: Amount which will be converted
- *Unit*: Source currency
- *Date*: Determines the date of conversion
- *Target Unit*: Defines the target currency
- *Timestamp*: Specifies the time stamp of processing

Output Fields:

- *Flow Value*: Converted amount
- *Flow Unit*: Target currency

The *Conversion* function uses exchange rates factors from standard SAP tables which are available on the Netweaver instance where SAP Profitability and Performance Management is deployed. For currency conversion, it uses the following tables:

- `TCURR`: Stores exchange rate data
- `TCURV`: Stores exchange rate types for currency translation
- `TCURX`: Stores decimal places in currency data
- `TCURN`: Stores quotation data

Procedure

Function Access

Follow the steps below to access the *Conversion* function of type “Currency Conversion”:

1. In the client where SAP Profitability and Performance Management is installed, choose *SAP Menu* > *Profitability and Performance Management* > *Modeling* > *Start My Environments*.
2. The *Environment* screen appears in a separate browser window. Choose an existing environment and continue. Within the environment, you can set up the newly added function.

Function Configuration

Follow the steps below to configure the *Conversion* function of type “Currency Conversion”:

1. In edit mode, configure the following required fields in the header. For more information about the header, see the *Key Features* section in Conversion [page 181].
   - *Result Handling*
   - *Suppress initial Result*

- *Include original Input Data*
- *Result Model Table*

2. Define the input function to be used on the *Input* tab.

> **i Note**
>
> For more information about the *Input* tab, see Input [page 11].

3. Define the fields to be used on the *Signature* tab.

> **i Note**
>
> For more information about the *Signature* tab, see Signature [page 13].

4. On the *Rules* tab, you can define the type of conversion (currency conversion or unit conversion) by defining the line type within the rule.
   Proceed as follows to configure the rules:
   1. On the *Rule* tab of your *Conversion* function, choose + (*Add*).
   2. The *Add Details* screen is displayed. Enter the following information:
      - *Add Level*
      - *Rule*
      - *Rule Type* = “Currency/Unit Conversion”
      - *Description*
   3. Choose *OK*.
   4. Select the created rule.
   5. On the *Rule Lines* tab of the created rule, choose + (*Add*).
   6. On the *Add Details* screen that is displayed next, enter the following information:
      - *Line*
      - *Line Type* = “Currency”
      - *Description*
   7. Choose *OK*.
   8. Select the created rule line.
   9. In the *Input Fields* section of the created line, enter the following information:
      - *Conversion Type*
      - *Value*
      - *Unit*
      - *Date*
      - *Target Unit*
      - *Timestamp*
   10. In the *Output Fields* section of the created line, enter the following information:
      - *Flow Value*
      - *Flow Unit*

5. Define the checks to be used on the *Checks* tab, if needed.

> **i Note**
>
> For more information about the *Checks* tab, see Checks [page 17].

6. Choose *Save* and then choose *Activate*.

Related Information

- For more information about common aspects of SAP Profitability and Performance Management functions, see Concepts for Key Users [page 6].
- For more information on how to add and remove functions in SAP Profitability and Performance Management environments, see Function Hierarchy [page 63].

1.4.2.4.1.1 Example: Currency Conversion

In this scenario, the source currency will be converted from source currency (USD) to target currency (GBP).

Input Data

Company Code	Value Data	Source Currency	Unit	Target Currency	Unit
CC01	20010101	100.00	USD	0	GBP

Table TCURR

Client	ExRt	From	To	Valid From	Exchange Rate
800	M	USD	GBP	20010101	2.000–

Given the exchange rate (USD to GBP) = 2.000– the amount in the target currency is calculated as follows: (which basically has a (–) negative notation):

Target Currency = Source Currency / 2.0 = 1.00 / 2.000 = 0.5

Therefore, USD 1 = GBP 0.500 and USD 100.00 = GBP 50.00.

Result

Company Code	Value Data	Source Currency	Unit	Target Currency	Unit
CC01	20010101	100.00	USD	50.00	GBP

1.4.2.4.2 Unit Conversion

The *Unit Conversion* function can convert between different units of measurement for the same quantity based on the conversion factors. Conversion factors are used to change the unit of a measured quantity without changing its value.

Input Fields:

- *Conversion Type*: Defines the conversion of unit using a predefined conversion table (T006)
- *Value*: Value which will be converted

- *Unit*: Source unit
- *Target Unit*: Defines the target unit

Output Fields:

- *Flow Value*: Converted value
- *Flow Unit*: Target unit

The *Conversion* function uses conversion factors from standard SAP tables which are available on the Netweaver instance where SAP Profitability and Performance Management is deployed. For unit conversion, it uses the following tables:

- `TCURF`: Stores conversion factor data
- `T006`: Stores units of measurement data
- `T006D`: Stores dimension data

Procedure

Function Access

Follow the steps below to access the *Conversion* function of type "Unit Conversion":

1. In the client where SAP Profitability and Performance Management is installed, choose *SAP Menu* > *Profitability and Performance Management* > *Modeling* > *Start My Environments*.
2. The *Environment* screen appears in a separate browser window. Choose an existing environment and continue. Within the environment, you can set up the newly added function.

Function Configuration

Follow the steps below to configure the *Conversion* function of type "Unit Conversion":

1. In edit mode, configure the following required fields in the header. For more information about the header, see the *Key Features* section in Conversion [page 181].
   - *Result Handling*
   - *Suppress initial Result*
   - *Include original Input Data*
   - *Result Model Table*
2. Define the input function to be used on the *Input* tab.

   > **i Note**
   >
   > For more information about the *Input* tab, see Input [page 11].
3. Define the fields to be used on the *Signature* tab.

   > **i Note**
   >
   > For more information about the *Signature* tab, see Signature [page 13].
4. On the *Rules* tab, you can define the type of conversion (currency conversion or unit conversion) by defining the line type within the rule.
   Proceed as follows to configure the rules:

1. On the *Rule* tab of your *Conversion* function, choose + (*Add*).
2. The *Add Details* screen is displayed. Enter the following information:
    - *Add Level*
    - *Rule*
    - *Rule Type* = “Currency/Unit Conversion”
    - *Description*
3. Choose *OK*.
4. Select the created rule.
5. On the *Rule Lines* tab of the created rule, choose + (*Add*).
6. On the *Add Details* screen that is displayed next, enter the following information:
    - *Line*
    - *Line Type* = “Unit”
    - *Description*
7. Choose *OK*.
8. Select the created rule line.
9. In the *Input Fields* section of the created line, enter the following information:
    - *Conversion Type*
    - *Value*
    - *Unit*
    - *Target Unit*
10. In the *Output Fields* section of the created line, enter the following information:
    - *Flow Value*
    - *Flow Unit*
5. Define the checks to be used on the *Checks* tab, if needed.

> **i Note**
>
> For more information about the *Checks* tab, see Checks [page 17].

6. Choose *Save* and then choose *Activate*.

Related Information

- For more information about common aspects of SAP Profitability and Performance Management functions, see Concepts for Key Users [page 6].
- For more information on how to add and remove functions in SAP Profitability and Performance Management environments, see Function Hierarchy [page 63].

1.4.2.4.2.1 Example: Unit Conversion

In this scenario, the distance from point A to point B is 519 km and will be converted from unit “KM” to target unit “CM”.

Input Data

Point A	Point B	Distance	Unit	Target Distance	Target Unit
AMSTERDAM	WALLDORF	519	KM	0	CM

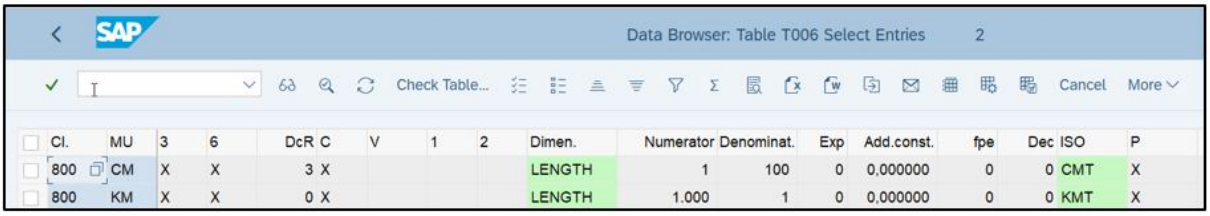


Table T006

Based on the values above, the function converts the unit from KM to CM as follows:

Target Distance = Distance * Numerator * Denominator = 519 * 1 000 = 100 = 51 900 000

Result

Point A	Point B	Distance	Unit	Target Distance	Target Unit
AMSTERDAM	WALLDORF	510	KM	51 900 000	CM

1.4.2.5 Derivation

Derivation is a data enrichment function that can be used to enhance the data in a dataset with calculated attributes based on predefined rules at runtime. The enriched data can then be used for consumption in downstream processes such as allocation. If the data to be derived is already available in the source data, the derived data is only overwritten if the condition values are met. Otherwise, the source values are retained.

Key Features

Header

In the header, you define the principal behavior of the derivation.

You can use the *Ensure Distinct Result* option with the following settings:

- Yes: Only the first successful derivation of overlapping derivation rules is included in the result and all subsequent matching derivations are excluded.
- No: All matching derivations of overlapping derivation rules are included in the result. This can lead to more result records than originally input.

Other header options, for example *Include Original Input Data*, *Result Handling*, *Suppress Initial Results* and *Result Model Table* are functional building blocks and are not specific to conversion. For more information about these options, see Header [page 10].

Rules

Each derivation rule semantically defines an if-then-statement. The if-part is maintained in the *Selection* section of a rule and the then-part in the *Action* section of a rule. In the if-part (*Selection* section), you specify

which subset of the input data the rule applies to. In the then-part (*Action* section), all fields specified are then filled with configured or set values.

Procedure

Function Access

Follow the steps below to access the *Derivation* function:

1. In the client where SAP Profitability and Performance Management is installed, choose *SAP Menu* > *Profitability and Performance Management* > *Modeling* > *Start My Environments*.
2. The *Environment* screen appears in a separate browser window. Choose an existing environment and continue. Within the environment, you can set up the newly added function.

Function Configuration

Follow the steps below to configure the *Derivation* function:

1. In edit mode, configure the following required fields in the header. For more information about the header, see the *Key Features* section above.
   - *Result Handling*
   - *Suppress initial Result*
   - *Ensure Distinct Result*
   - *Include original Input Data*
   - *Result Model Table*
2. Define the input function to be used on the *Input* tab.

   > **i Note**
   >
   > For more information about the *Input* tab, see Input [page 11].

3. Define the fields to be used on the *Signature* tab.

   > **i Note**
   >
   > For more information about the *Signature* tab, see Signature [page 13].

4. On the *Rules* tab, each derivation rule semantically defines an if-then-statement. In the “if”-part (“Selection” section of the rule), you specify which subset of the input data the rule applies to. In the “then”-part (*Action* section of the rule), all fields specified are then filled with configured or set values. Proceed as follows to configure the rules:
   1. On the *Rule* tab of your *Derivation* function, choose + (*Add*).
   2. The *Add Details* screen is displayed. Enter the following information:
      - *Rule*
      - *Description*
   3. Choose *OK*.
   4. Select the created rule.
   5. In the *Selection* section of the created rule, choose + (*Add*).
   6. On the *Fields* screen that is displayed next, choose the field to be used and then choose *OK*.

7. Select the field and choose *Selection Condition*.
8. On the *Selection Condition* screen, define the selection condition to be used and choose *OK*.
9. In the *Action* section of the created rule, choose + (*Add*).
10. On the *Fields* screen that is displayed next, choose the field to be used and choose *OK*.
11. Select the field and choose *Formula*.
12. On the *Formula* screen that is displayed next, define the formula to be used and choose *OK*.

5. Define the checks to be used on the *Checks* tab, if needed.

> **i Note**
>
> For more information about the *Checks* tab, see Checks [page 17].

6. Choose *Save* and then choose *Activate*.

Related Information

- For more information about the configuration of the *Derivation* function, see How to Configure a Derivation Function.
- For more information about common aspects of SAP Profitability and Performance Management functions, see Concepts for Key Users [page 6].
- For more information on how to add and remove functions in SAP Profitability and Performance Management environments, see Function Hierarchy [page 63].

1.4.2.5.1 Example: Simple Condition

This scenario is meant to test a simple derivation. The processed data (*Include Original Input Data* = "No") which are shown here only contain the records which satisfy the rule condition (*Result Handling* = "Include enriched data").

Input

Contract	Product 1	Origination Date	Amount	Premium
SUNSHINE	LIFE1	2016-01-01	600	0
SUNSHINE	LIFE2	2016-01-10	400	0
SUNSHINE	LIFE3	2016-01-15	500	0
MOONLIGHT	NONLIFE1	2016-01-04	200	0
MOONLIGHT	NONLIFE2	2016-01-21	300	0
MOONLIGHT	NONLIFE3	2016-01-29	100	0

Applying the following rule, we will get the result from the table below:

```
If Contract = SUNSHINE and Product 1 = LIFE1 then Premium = Amount
```

Result

Contract	Product 1	Origination Date	Amount	Premium
SUNSHINE	LIFE1	2016-01-01	600	600

1.4.2.5.2 Example: Ensure Distinct Result

This scenario shows that if the record has already run through a rule, it is no longer considered and will be ignored already in the next rule even if it satisfies the condition.

Input: Customer-Branch Table

Branch	Customer	Customer Type	Deposit Amount	Interest Rate
B1	C1	New	5,000	0.01
B2	C2	Regular	10,000	0.015
B3	C3	Loyal	15,000	0.02
B4	C4	Regular	50,000	0.015
B1	C5	Regular	45,000	0.015
B1	C6	Loyal	120,000	0.02
B2	C7	Loyal	56,000	0.02
B3	C8	Loyal	70,000	0.02
B2	C9	New	105,000	0.01
B4	C10	New	80,000	0.01
B4	C11	Regular	60,000	0.015
B1	C12	Loyal	80,000	0.02

The system applies the following rule to derive the loyal customer from B1, and adds additional interest respectively to the derived values:

```
If Branch = B1 and Customer Type = LOYAL then Additional Interest = 0.0500
```

The deposit with interest is computed for the interim result:

Interim Result: First Rule

Branch	Customer	Customer Type	Deposit Amount	Interest Rate	Additional Interest	Deposit with Interest
B1	C6	Loyal	120,000	0.02	0.5	123,000
B1	C12	Loyal	80,000	0.02	0.5	82,000

Afterwards, the system applies the following rule:

```
If Customer Type = LOYAL and Deposit Amount > 50,000 then Additional Interest =
0.0250
```

Since B1 has already been derived, the system no longer includes it in the derivation. The system adds the additional interest respectively to the derived values and computes the deposit with interest:

Interim Result: Second Rule

Branch	Customer	Customer Type	Deposit Amount	Interest Rate	Additional Interest	Deposit with Interest
B2	C7	Loyal	56,000	0.02	0.25	57,260
B3	C8	Loyal	70,000	0.02	0.25	71,575

After the system has applied both rules, the final result is as follows:

Result

Branch	Customer	Customer Type	Deposit Amount	Interest Rate	Additional Interest	Deposit with Interest
B1	C6	Loyal	120,000	0.02	0.5	123,000
B1	C12	Loyal	80,000	0.02	0.5	82,000
B2	C7	Loyal	56,000	0.02	0.25	57,260
B3	C8	Loyal	70,000	0.02	0.25	71,575

1.4.2.6 Flow Modeling

The *Flow Modeling* function offers a set of rule types that provide different calculation logic to process different kinds of business requirements. Each rule type represents an independent and reusable logic for enriching amounts, factors and dates of cash flows consumed, for example. You can add several rule types to the same *Flow Modeling* function, which can run in parallel or sequentially. In sequential processing the successor rule type consumes the results from the predecessor rule type. You can also add parallel-processed line items for some rule types. The key features of each configuration are explained below.

Key Features

Rules

Flow Modeling consists of several rule types where each rule type represents an encapsulated and reusable logic for the calculation of data.

The following rule types and corresponding examples are available:

1. Formula [page 195]:
   Applies formulas and SQL functions to characteristics and key figures.
2. Flow Cut-off [page 197]:
   Applies a cut-off to cash flow data according to a given reference day. The rule type deletes all cash flow items prior to the cut-off period.
3. Series Generation [page 200]:
   Generates series data by providing several parameters such as *Step Size*, *Series Type*, *Period From* and *Period To*..

4. Term Conversion [page 203]:
   Converts terms of different periodicities into a common basis of days. The rule type offers the option to choose different day count conventions (30/360 German, ACT/ACT).
5. Term Selection [page 205]:
   Selects a specific term on a month-end basis from a cash flow taking into consideration which period type (monthly, quarterly, yearly) and day count convention (30/360 German, ACT/ACT) is set in the configuration.
6. Term Target [page 208]:
   Enriches a given set of cash flow data (based on a daily periodicity) and returns a cash flow structure that uses a consistent periodicity of months, quarters or years, and that can be based on different sorts of day count conventions (30/360 German, ACT/ACT). It also interpolates the values of the terms added to the original pattern structure.
7. Term To Date [page 212]:
   Converts a given set of terms of cash flows into dates referencing a given start date. The configuration allows you to choose between a default approach and an approach that applies different logic to distinguish between pattern items which are of balance type (cumulative factor/amount values) or movement type (delta factors/amounts).
8. Value Conversion [page 216]:
   Comprises two different calculation methods that can be used either to sum up cash flow items over a given set of terms (running total), or to calculate the delta values between a given set of cash flow items (balance = cumulative values, movement = delta values).
9. Incremental Value Calculation [page 218]:
   Combines two factor patterns and computes a mutual pattern.
10. Redistribution [page 223]:
    Calculates estimate values prior to the Reference Date for Redistribution (RDR) and redistributes them to the future periods after this date.
11. Scale Factor [page 227]:
    Ratio of two corresponding values with similar field/data types (division).
12. Scaling [page 228]:
    Applies a scale factor to the actuarial model stream (multiplication).
13. Acknowledge Actuals [page 230] (Acknowledgement of Cedent Data):
    Enriches the actuals by determining the missing earlier life cycle date information by applying matching logic.
    The matching of the actuals to the estimates can be carried out in the following ways:
    - If the CF calculation is "01", the system matches the actual to the estimate based on the business date of the actual. In other words, *Settled Date* for settled transactions, *Due Date* for due transactions, and *Reported Date* for reported actuals.
    - If the CF calculation is "02", the system matches the actual to the estimates based only on the Secondary Risk Incurred Date. This is applied in L&H business where the missing dates in the actuals are predetermined by a reverse life cycle conversion based on the models.
14. Life Cycle Conversion [page 233]:
    Applies the lags and lag factors delivered by the actuarial input in the form of a lag factor pattern to the cashflow stream to determine the amounts and date for the lifecycle stage.
15. Modulation In [page 236]:
    - New contract:
      Modulation In for contract T must be equal to Modulation Out for contract T multiplied by –1.
    - Renewed contract:
      Modulation In for contract T must be equal to Modulation Out for contract T–1 multiplied by –1.

- Incurred date of Modulation In cash flows should be set to be in the first period of contract T.

16. Modulation Out [page 238]:
    - Selects amounts in the basis cash flow where the exposure date > coverage end date.
    - If Modulation Out for contract T is updated and contract T+1 exists, Modulation In for contract T+1 must be (re-)calculated.
17. Item Number Generation [page 239]:
    Separates each partition by creating a number for each one. To do this, the system needs a *Granularity* field (which separates the partitions from each other) and an *Item Number* field (which ise filled by this rule type).
18. Cashflow Regime [page 241]:
    Adjusts the cashflow stream based on the regime into which each of the cashflows falls:
    - Follow Actuals (01)
      This regime comprises only the effect of actuals. Therefore the system removes any estimate item that falls in this regime from the final cashflow.
    - Reflect Actuals (02)
      This regime comprises only the effect of actuals. Model-based estimates do not have any effect in this regime. However, more actuals may be expected to be reported in this period. The system therefore calculates an additional incurred estimate as a factor of the actuals. These additional Incurred estimates will have the same date information as that of the actuals but the amounts will be calculated as a factor of the actuals and will apply the formula (Amount * Factor / (1 - Factor)).
    - Follow Estimates (03)
      In this regime, the model-based estimates are expected to be effective. Therefore, for every actual that falls into this regime an additional negated estimate is introduced into the cashflow with the same date information as the actuals.
    - Estimated Future (04)
      In this regime, no actual information is expected.
19. Clear Actual Dates [page 246]
    Clears the actual date information in the cashflows arising from actuals.
20. Incurred to Reported Factor Calculation [page 248]
    Calculates the *Incurred to Reported* lags in cases where these lags are not delivered directly, using the more granular *Policy Holder to Primary Insurer* lag and *Primary Insurer to Reinsurer* lag.
21. Factor for Additional Incurred Cash Flow [page 251]
    Calculates the factors to be applied in the Reflect Actual Regime based on the delivered *Policy Holder to Primary Insurer* lags. The factors are calculated as a difference between 1 and the cumulated sum of the policy holder to primary insurer lag factors for a certain granularity. The number of periods is determined by the Reflect Actuals attachment point.

Hierarchical rules are supported by assigning higher levels. In this case, the hierarchy of the levels is resolved starting with the lowest level, which is fed as input to the higher levels, and ending with the highest level.

Sub View

You can define further selections, aggregations and sorting orders for each rule.

Procedure

Function Access

Follow the steps below to access the *Flow Modeling* function:

1. In the client where SAP Profitability and Performance Management is installed, choose *SAP Menu* > *Profitability and Performance Management* > *Modeling* > *Start My Environments* .
2. The *Environment* screen appears in a separate browser window. Choose an existing environment and continue. Within the environment, you can set up the newly added function.

Function Configuration

Follow the steps below to configure the *Flow Modeling* function:

1. In edit mode, configure the following required fields in the header.
   - *Result Handling*
   - *Suppress initial Result*
   - *Include original Input Data*
   - *Result Model Table*
2. Define the input function to be used on the *Input* tab.

   > **i Note**
   >
   > For more information about the *Input* tab, see Input [page 11].
3. Define the function to be used on the *Lookup* tab, if needed.

   > **i Note**
   >
   > For more information about the *Lookup* tab, see Lookup [page 12].
4. Define the fields to be used on the *Signature* tab.

   > **i Note**
   >
   > For more information about the *Signature* tab, see Signature [page 13].
5. On the *Rules* tab, each rule type represents an encapsulated and reusable logic for the calculation of data. For more information about the available rule types, see the *Key Features* section above.
   Proceed as follows to configure the rules:
   1. On the *Rule* tab of your *Flow Modeling* function, choose + (*Add*).
   2. The *Add Details* screen is displayed. Enter the following information:
      - *Add Level*
      - *Rule*
      - *Rule Type*
      - *Description*
   3. Choose *OK*.
   4. Select the created rule.
   5. On the *Rule Lines* tab of the created rule, choose + (*Add*).
   6. The *Add Details* screen is displayed. Enter the required information.

7. Select the created rule line and enter the required information.

> **i Note**
>
> Depending on the chosen rule type the rule line shows different fields.

6. Define the checks to be used on the *Checks* tab, if needed.

> **i Note**
>
> For more information about the *Checks* tab, see Checks [page 17].

7. Choose *Save* and then choose *Activate*.

Related Information

- For more information about common aspects of SAP Profitability and Performance Management functions, see Concepts for Key Users [page 6].
- For more information on how to add and remove functions in SAP Profitability and Performance Management environments, see Function Hierarchy [page 63].

1.4.2.6.1 Example: Formula

Applying Formulas and SQL Functions to Characteristics

In the example, we apply a simple SQL formula to compare two fields containing dates and we mark a third field according to that formula.

Input Data

Contract ID	Date_1	Date_2	Total Premium
A	2019-01-01	2019-02-01	300
B	2019-04-01	2019-03-01	400

Flow Modeling Configuration

Rules (Tab)

Rule	Description	Rule Type	State	Rule Grouping	Rule Ordering	Selection
CHARF	Characteristic Formula	Characteristic Formula	Active			

Rule Line

Line	Formula	Result

L001	`CASE`
	`WHEN DATE_1 > DATE 2`
	`THEN 'X'`
	`ELSE "`
	`END`
		`DATE_IND`

When *DATE_1* is greater than *DATE_2*, the field *DATE_IND* (Date Indicator) is marked with “X”.

Key Configuration Description

The table below explains the meaning of the options required to run this configuration.

Field	Meaning	Notes
Line	Identifier of the line	Must be unique. A single characteristic formula rule can run multiple lines.
Formula	SQL formula that has to be applied to categorical variables	Mandatory input
Result	Field where the result of the rule is stored	Mandatory output. Must be defined in the *Action* section of the *Signature* tag.

Expected Result

Contract ID	Date_1	Date_2	Total Premium	Date Indicator
A	2019-01-01	2019-02-01	300	
B	2019-04-01	2019-03-01	400	X

Applying Formulas and SQL Functions to Key Figures

In this example, we apply a very simple SQL formula to compute the weighted premium (`WEIGHTED_PREMIUM`) based on the fields `TOTAL_PREMIUM` and `WEIGHT`.

Input Data

Contract ID	Date_1	Total Premium	Weight
A	2019-01-01	300	0.4
B	2019-04-01	400	0.6

Flow Modeling Configuration

Rules (Tab)

Rule	Description	Rule Type	State	Rule Grouping	Rule Ordering	Selection
KFF	Key Figure Formula	Formula	Active			

Rule Line

Line	Formula	Result
L001	`TOTAL_PREMIUM*WEIGHT`	`WEIGHTED_PREMIUM`

Key Configuration Description

The table below explains the meaning of the options required to run this configuration.

Field	Meaning	Notes
Line	Identifier of the line	Must be unique. A single *Key Figure Formula* rule can run multiple lines.
Formula	SQL formula that has to be applied to numerical variable	Mandatory input
Result	Field where the result of the rule is stored	Mandatory output. Must be defined in the *Action* section of the *Signature* tag.

Expected Output

Contract ID	Date_1	Total Premium	Weight	Weighted Premium
A	2019-01-01	300	0.4	120
B	2019-04-01	400	0.6	240

1.4.2.6.2 Example: Flow Cut-Off

Applies a cut-off to cash flow data according to a given reference day. The rule type deletes all cash flow items prior to the cut-off period.

This function gives two different results depending on the type of input values: for delta values (MOV values), the pattern after the cut-off starts from period one; for cumulative values (BAL values), the resulting pattern starts from zero. The function can handle the day count conventions 30/360 German and ACT/ACT. Two different examples are shown below.

Input Data

Start Date	Contract ID	Pattern KF Type	Period Unit	Period To	Result Value	Cut-Off Period
2019-01-01	A	MOV	4	30	EUR 30	60
2019-01-01	A	MOV	4	60	EUR 60	60
2019-01-01	A	MOV	4	90	EUR 90	60
2018-01-01	A	MOV	4	120	EUR 120	60
2019-01-01	B	MOV	4	30	EUR 30	30
2019-01-01	B	MOV	4	60	EUR 60	30
2019-01-01	B	MOV	4	90	EUR 90	30
2019-01-01	C	MOV	4	30	EUR 30	30
2019-01-01	C	MOV	4	60	EUR 60	30
2019-01-01	C	MOV	4	90	EUR 90	30
2019-01-01	C	MOV	4	120	EUR 120	30

In the table above, there are three patterns for three contracts. The field `PERIOD_UNIT` is not used in the configuration. It refers to `PERIOD_UNIT` and`PERIOD_TO` and indicates that they are days.

Flow Modeling Configuration

Rules (Tab)

Rule	Description	Rule Type	State	Rule Grouping	Rule Ordering	Selection
FCF	Flow Cut-Off	Flow Cut-Off	Active			

Rule Line

Line	Line Granularity	Day Count Convention	Start Date	Value Type	Period To	Cut-Off Comparison	Period To Result	Period Unit
L001	`CONTRACT_ID`	German 30/360	`START_DATE`	`PATTERN_KF_TYPE`	`PERIOD_TO`	`CUT_OFF_PERIOD`	`PERIOD_TO_RESULT`	`PERIOD_UNIT_RESULT`

Key Configuration Description

Field	Meaning	Notes
Line	Identifier of the line.	Must be unique.

Field	Meaning	Notes
Line Granularity	Determines the size of one partition of data.	Mandatory input that has to be inserted in the *Granularity* section of the *Signature* tag.
Day Count Convention	Day count convention used to determine the result periods.	Mandatory input. The conventions managed by the function are: • Actual/Actual • German 30/360
Start Date	Starting date to compute the day count.	Mandatory input.
Value Type	You can select the value "MOV" or "BAL" to determine which value type the cash flow item is made of.	Mandatory input. The user has to make sure that the field contains these two values only.
Period To	Number of days of the period.	Mandatory input
Cut-Off Comparison	Key field to identify which cash flow items need to be deleted.	Mandatory input Rule in case of delta items: Cut all periods <= cut-off period. If the balance is not zero, the system treats the cut-off period as a new period.
Period To Result	Output field to indicate the specific period of pattern of the cut-off.	Mandatory output
Period Unit Result	Value type of the period result.	Mandatory output. The value "6" stands for "month"

Expected Output

Start Date	Contract ID	Pattern KF Type	Period To	Result Value	Cut-Off Period	Period To Result	Period Unit Result
2019-01-01	A	MOV	90	EUR 90	60	**1**	**6**
2019-01-01	A	MOV	120	EUR 120	60	**2**	**6**
2019-01-01	B	MOV	60	EUR 60	30	**1**	**6**
2019-01-01	B	MOV	90	EUR 90	30	**2**	**6**
2019-01-01	C	MOV	60	EUR 60	30	**1**	**6**
2019-01-01	C	MOV	90	EUR 90	30	**2**	**6**
2019-01-01	C	MOV	120	EUR 120	30	**3**	**6**

All cash flow items prior to or equal to the cut-off period have been deleted.

Example with BAL Data

The configuration does not change with respect to the previous example. What is different now is the value of the field `PATTERN_KF_TYPE` (moreover, the field `PERIOD_UNIT` has been deleted).

Input Data

Start Date	Contract ID	Pattern KF Type	Period To	Result Value	Cut-Off Period
2019-01-01	A	BAL	30	EUR 30	60
2019-01-01	A	BAL	60	EUR 60	60
2019-01-01	A	BAL	90	EUR 90	60
2019-01-01	A	BAL	120	EUR 120	60

Output Data

Start Date	Contract ID	Pattern KF Type	Period To	Result Value	Cut-Off Period	Period To Result	Period Unit Result
2019-01-01	A	BAL	60	EUR 60	60	0	6
2019-01-01	A	BAL	90	EUR 90	60	1	6
2019-01-01	A	BAL	120	EUR 120	60	2	6

1.4.2.6.3 Example: Series Generation

Generates series data by providing several parameters, like *Step Size*, *Series Type*, *Period From* and *Period To*.

Input Data

Contract ID	Period From	Period To
A	1	3
A	4	4

Contract ID	Period From	Period To
B	1	2
B	3	8

Contract ID	Period From	Period To
A	1	3

Contract ID	Period From	Period To
A	1	2

Contract ID	Period From	Period To
A	3	8

Flow Modeling Configuration

Rules (Tab)

Rule	Description	Rule Type	State	Rule Grouping	Rule Ordering	Selection
GS	Generate Series	Generate Series	Active			CON-TRACT="A"

Rule Line

Line	Incre-mented By	Minimum	Maximum	Series Type	Element Number	Fraction	Period From	Period To Result
1	1	`PERIOD_FROM`	`PERIOD_TO`	Integer	`PERIOD_NUMBER`			

> **i** Note
>
> Condition = Filter options that select only cash flow items containing filter option values. In this example the rule is computing only for cash flow items which have a value of "A" in column CONTRACT.

Key Configuration Description

> **i** Note
>
> This rule type makes use of the SAP HANA SQL function `SERIES_GENERATE`. For more information about this function, see SAP HANA SQL Reference Guide for SAP HANA Platform SERIES_GENERATE Function (Series Data).

Field	Meaning	Notes
Line	Identifier of the line	Must be unique.
Increment by	Field or single value that defines the number added to the value of the field *Minimum*.	Mandatory input
Minimum	Starting value of the series	Mandatory input
Maximum	Ending value of the series	Mandatory input

Field	Meaning	Notes
Series Type	Defines the type of input and affects the resulting output.	Mandatory input. Possible values: • Integer • Date • Decimal
Element Number	The system writes the main result of the function to this fiedl.	Mandatory output. You need to insert the target field in the *Action* tag of the *Signature* section.
Fraction	This field can be used to save an additional result of the function.	The following calculation is computed: Minimum value of the range/maximum value of the range.
Period From Result	This field can be used to save an additional result of the function.	Period From + Increment by
Period To Result	This field can be used to save an additional result of the function.	Period From Result + Increment by

Caution

Depending on your individual data, the *Series Generation* rule type can add up to several hundreds or even thousands of new lines for each entry. This can have a significant impact on the performance. Using the general configuration and assigning input fields to the mapping fields (*Increment by*, *Minimum* and *Maximum*) allows you to process only a few hundreds of records. However, assigning constants to all three mapping fields instead allows you to process a significantly higher number of records.

Expected Result

Contract ID	Period From	Period To	Period Number
A	1	3	**1**
A	1	3	**2**
A	1	3	**3**
A	4	4	**1**

Contract ID	Period From	Period To	Period Number
A	1	3	**1**
A	1	3	**2**
A	1	3	**3**

Contract ID	Period From	Period To	Period Number
A	1	2	**1**
A	1	2	**2**
A	3	8	**1**
A	3	8	**2**
A	3	8	**3**
A	3	8	**4**
A	3	8	**5**
A	3	8	**6**

The three patterns are shown separately for clarity of exposition.

1.4.2.6.4 Example: Term Conversion

Converts terms of different periodicities into a common basis of days. The rule type offers the option to choose different day count conventions (30/360 German, ACT/ACT).

In this example, we show two patterns in the input data: a pattern for contract A and a pattern for contract B. We filter for contract A using a subview. The day conversion is performed starting from the key date on the field inserted in *Period To* on the *Configuration* tab. The result is a range in days.

Input Data

The fields *Pattern Key Figure Type* and *Value* are not relevant for the conversion.

Start Date	Contract ID	Pattern KF Type	Period To	Cal FreqCode	Value
2019-01-01	A	MOV	3	6	EUR 90
2019-01-01	A	MOV	4	6	EUR 120
2019-01-01	A	MOV	5	6	EUR 150
2019-01-01	A	MOV	6	6	EUR 180

Start Date	Contract ID	Pattern KF Type	Period To	Cal FreqCode	Value
2019-01-16	B	MOV	2	6	EUR 50
2019-01-16	B	MOV	8	6	EUR 150

Flow Modeling Configuration

Rules (Tab)

Rule	Description	Rule Type	State	Rule Grouping	Rule Ordering	Selection
TC	Term Conversion	Term Conversion	Active			CONTRACT = “A”

Rule Line

Line	Line Granularity	Start Date	Day Count Convention	Period To	Period Unit	Period To Conv	Period From Conv	Period Unit Conv
1	Contract ID	Start Date	German 30/360	Period	Period Unit	Period to Converted	Period from Converted	Period Unit Converted

> **i** Note
>
> Condition = Filter out options that select only cash flow items containing filter option values. In this example, the rule computes only for cash flow items with an “A” entry in the *Contract ID* column.

Key Configuration Description

Field	Meaning	Notes
Line	Identifier of the line	Must be unique.
Line Granularity	Determines the size of one partition of data	Mandatory input that has to be inserted in the *Granularity* tab of the *Signature* section.
Start Date	Starting date to compute the day count	Mandatory input
Day Count Convention	Day count convention used to determine the result period	Mandatory input. The conventions managed by the function are: • US 30/360 • Actual/Actual • EU 30/360 • German 30/360
Period To	Parameter to indicate the period to be added to *Start Date*	Mandatory input

Field	Meaning	Notes
Period Unit	Period type of input data (for example, monthly or quarterly)	Mandatory input. The possible values are the following: • 4 if *Period To* is in days • 5 if *Period To* is in weeks • 6 if *Period To* is in months • 11 if *Period To* indicates quarters • 12 if *Period To* indicates half a year • 7 if *Period To* indicates years
Period From Conv	Exact day from which the period count starts.	Mandatory output
Period To Conv	Number of days of the period	Mandatory output
Period Unit Conv	The system writes the period unit of the result to this field.	Mandatory output. The function performs the conversion from the number in the field inserted in *Period Unit* to days (value equal to 4).

Expected Result

Start Date	Contract ID	Pattern KF Type	Period To	Cal Freq Code	Value	Period From Conv	Period To Conv	Period Unit Converted
2019-01-01	A	MOV	3	6	EUR 90	**1**	**90**	**4**
2019-01-01	A	MOV	4	6	EUR 120	**91**	**120**	**4**
2019-01-01	A	MOV	5	6	EUR 150	**121**	**150**	**4**
2019-01-01	A	MOV	6	6	EUR 180	**151**	**180**	**4**

1.4.2.6.5 Example: Term Selection

Selects a specific term on a month-end basis from a cash flow taking into consideration which period type (monthly, quarterly, yearly) and day count convention (30/360 German, ACT/ACT) is set in the configuration.

In the example for this type of configuration, we run the function for three different patterns and for two types of target conversion (from days to months and from days to quarters). The output is a new pattern for the three contract IDs with a new periodicity.

Input Data

Start Date	Contract ID	Pattern KF Type	Period Unit	Period To	Result Value
2019-01-01	A	MOV (Delta)	3	30	EUR 30
2019-01-01	A	MOV (Delta)	4	60	EUR 60
2019-01-01	A	MOV (Delta)	2	90	EUR 90
2019-01-01	A	MOV (Delta)	8	120	EUR 120

Start Date	Contract ID	Pattern KF Type	Period Unit	Period To	Result Value
2019-01-01	B	MOV (Delta)	4	30	EUR 30
2019-01-01	B	MOV (Delta)	4	60	EUR 60
2019-01-01	B	MOV (Delta)	4	90	EUR 90

Start Date	Contract ID	Pattern KF Type	Period Unit	Period To	Result Value
2019-01-01	C	MOV (Delta)	4	30	EUR 30
2019-01-01	C	MOV (Delta)	4	60	EUR 60
2019-01-01	C	MOV (Delta)	4	90	EUR 90
2019-01-01	C	MOV (Delta)	4	120	EUR 120

Flow Modeling Configuration

Rules (Tab)

Rule	Description	Rule Type	State	Rule Grouping	Rule Ordering	Selection
TS	Term Selection	Term Selection	Active			

Rule Line

Line	Line Granularity	Day Count Convention	Start Date	Period Type	Value Type	Period To	Period To Result	Period Unit Result
1	Contract ID	German 30/60	Start Date	Monthly/ Quarterly		Period To	Period To Result	Period Unit Result

Key Configuration Description

Field	Meaning	Notes
Line	Identifier of the line	Must be unique

Field	Meaning	Notes
Line Granularity	Determines the size of one partition of data	Mandatory input that has to be inserted in the *Granularity* tab of the *Signature* section
Day Count Convention	Day count convention used to determine the result period	Mandatory input. The function manages the following conventions : • US 30/360 • Actual/Actual • EU 30/360 • German 30/360
Start Date	Starting date to compute the day count	Mandatory input
Period Type	Determines the final pattern structure (for example, monthly or quarterly).	Mandatory input
Period To	Day count of the starting pattern for the cash flow	
Period To Result	The system writes the period type of the output to this field.	Mandatory output
Period Unit Result	Period type of output data (for example, monthly or quarterly)	Mandatory input depending on the value that has been defined under *Period Type*. The following values are possible: • 6 if *Period To* is in months • 11 if *Period To* indicates quarters • 7 if *Period To* indicates years

Expected Output

Expected Output for Period Type = Monthly

Start Date	Contract ID	Pattern KF Type	Period To	Period Unit	Period To Result
2019-01-01	A	MOV (Delta)	30	6	1
2019-01-01	A	MOV (Delta)	60	6	2
2019-01-01	A	MOV (Delta)	90	6	3
2019-01-01	A	MOV (Delta)	120	6	4

Start Date	Contract ID	Pattern KF Type	Period To	Period Unit	Period To Result
2019-01-01	B	MOV (Delta)	30	6	**1**

Start Date	Contract ID	Pattern KF Type	Period To	Period Unit	Period To Result
2019-01-01	B	MOV (Delta)	60	6	**2**
2019-01-01	B	MOV (Delta)	90	6	**3**

Start Date	Contract ID	Pattern KF Type	Period To	Period Unit	Period To Result
2019-01-01	C	MOV (Delta)	30	6	1
2019-01-01	C	MOV (Delta)	60	6	2
2019-01-01	C	MOV (Delta)	90	6	3
2019-01-01	C	MOV (Delta)	120	6	4

Expected Output for Period Type = Quarterly

Start Date	Contract ID	Pattern Key Figure Type	Period To	Period Unit Result	Period To Result
2019-01-01	A	MOV (Delta)	90	11	1
2018-01-01	A	MOV (Delta)	120	11	2

Start Date	Contract ID	Pattern Key Figure Type	Period To	Period Unit Result	Period To Result
2019-01-01	B	MOV (Delta)	90	11	1

Start Date	Contract ID	Pattern Key Figure Type	Period To	Period Unit Result	Period To Result
2019-01-01	C	MOV (Delta)	90	11	1
2019-01-01	C	MOV (Delta)	120	11	2

1.4.2.6.6 Example: Term Target

Enriches a given set of cash flow data (based on a daily periodicity) and returns a cash flow structure that uses a consistent periodicity of months, quarters or years, and that can be based on different sorts of day count conventions (30/360 German, ACT/ACT). It also interpolates the values of the terms added to the original pattern structure.

In the example, we present a pattern for MOV values (delta values) that are broken into smaller periods. The function proportionally spreads the amount between the resulting ranges.

Input Data

Contract ID	Start Date	Pattern Key Figure Type	Value	Period From	Period To	Period Unit
A	2019-01-01	MOV (Delta)	EUR 90	1	90	4
A	2019-01-01	MOV (Delta)	EUR 120	91	120	4

Contract ID	Start Date	Pattern Key Figure Type	Value	Period From	Period To	Period Unit
B	2019-01-01	MOV (Delta)	EUR 90	1	90	4

Contract ID	Start Date	Pattern Key Figure Type	Value	Period From	Period To	Period Unit
C	2019-01-01	MOV (Delta)	EUR -	1	90	4
C	2019-01-01	MOV (Delta)	EUR 120	91	120	4

Flow Modeling Configuration

Rules (Tab)

Rule	Description	Rule Type	State	Rule Grouping	Rule Ordering	Selection
TT	Term Target	Term Target	Active			

Rule Line

Line	*Line Granularity	*Start Date	Cut-Off Comparison	*Value Type	*Day Count Convention	*Period Type	*Period From	*Period To	*Value (Result)	*Period To Result	*Period Unit Result	Period Cut-Off
1	Contract ID	Start Date	–	Pattern KF Type	German 30/60	Monthly	Period From	Period To	Value	Period To Result	Period Unit Result	

Key Configuration Description

Field	Meaning	Notes
Line	Identifier of the line	Must be unique.
Line Granularity	Determines the size of one partition of data	Mandatory input that has to be inserted in the *Granularity* section of the *Signature* tab
Start Date	Starting date to compute the day count	Mandatory input

Field	Meaning	Notes
Cut-Off Comparison	Cut-off date for a preparational step	All items prior to or equal to this date are deleted later if the rule type flow cut-off is also applied.
Value Type	This field can have the value "MOV" or "BAL" to determine which value type the cash flow item is made of	Mandatory input. Ensure that the field contains the values "MOV" or "BAL" only. These values differ in the way that the starts of the period are managed.
Day Count Convention	Day count convention used to determine the result period	Mandatory input. The funtion manages the following conventions : • US 30/360 • Actual/Actual • EU 30/360 • German 30/360
Period Type	Parameter to indicate the pattern structure of the output	Mandatory input. The possible values are: • Monthly • Quarterly • Yearly
Period From	Start of the range	Mandatory input
Value (Result)	This field is both the input field for the amount to be spread and the field used to store the result.	Mandatory input/output
Period To Result	New pattern structure	Mandatory output
Period Unit Result	Period unit of the pattern	Mandatory output depending on *Value Type*

Expected Output

Start Date	Contract ID	Pattern KF Type	Period From	Period To	Period Unit	Period Unit Result	Period To Result	Result Value
2019-01-01	A	MOV (Delta)	91	120	4	6	0	EUR 0
2019-01-01	A	MOV (Delta)	1	90	4	6	30	EUR 30
2019-01-01	A	MOV (Delta)	1	90	4	6	60	EUR 60

Start Date	Contract ID	Pattern KF Type	Period From	Period To	Period Unit	Period Unit Result	Period To Result	Result Value
2019-01-01	A	MOV (Delta)	1	90	4	6	90	EUR 90
2019-01-01	A	MOV (Delta)	91	120	4	4	120	EUR 120

Start Date	Contract ID	Pattern KF Type	Period From	Period To	Period Unit	Period Unit Result	Period To Result	Result Value
2019-01-01	B	MOV (Delta)	1	90	4	6	0	EUR 0
2019-01-01	B	MOV (Delta)	1	90	4	6	30	EUR 30
2019-01-01	B	MOV (Delta)	1	90	4	6	60	EUR 60
2019-01-01	B	MOV (Delta)	1	90	4	6	90	EUR 90

Start Date	Contract ID	Pattern KF Type	Period From	Period To	Period Unit	Period Unit Result	Period To Result	Result Value
2019-01-01	C	MOV (Delta)	91	120	4	6	0	EUR 0
2019-01-01	C	MOV (Delta)	1	90	4	6	30	EUR 30
2019-01-01	C	MOV (Delta)	1	90	4	6	60	EUR 60
2019-01-01	C	MOV (Delta)	1	90	4	6	90	EUR 90
2019-01-01	C	MOV (Delta)	91	120	4	4	120	EUR 120

If we change the *Period Type* from “Monthly” to “Quarterly” the result for contract A is as follows:

Contract ID	Start Date	Pattern KF Type	Period From	Period To	Period Unit	Value	Period To Result	Period Unit Result
A	2019-01-01	MOV	91	120	4	0	0	11
A	2019-01-01	MOV	1	9	4	90	90	11
A	2019-01-01	MOV	91	120	4	120	120	11

1.4.2.6.7 Example: Term To Date

Converts a given set of terms of cash flows into dates referencing a given start date.

The configuration allows you to choose between a default approach and an approach that applies different logic to distinguish between pattern items that are of either balance type (cumulative factor/amount values) or movement type (delta factors/amounts).

The examples below are both for patterns of type MOV (delta) and of type BAL (cumulative).

Input Data with Pattern MOV (Delta) Values

Start Date	Contract ID	Pattern KF Type	Period Unit	Period To
2019-01-01	A	MOV (Delta)	6	1
2019-01-01	A	MOV (Delta)	6	2

Start Date	Contract ID	Pattern KF Type	Period Unit	Period To
2019-01-06	B	MOV (Delta)	6	1
2019-01-06	B	MOV (Delta)	6	2

Start Date	Contract ID	Pattern KF Type	Period Unit	Period To
2019-01-01	C	MOV (Delta)	6	1
2019-01-01	C	MOV (Delta)	6	2
2019-01-01	C	MOV (Delta)	6	3
2019-01-01	C	MOV (Delta)	6	4

Flow Modeling Configuration

Rules (Tab)

Rule	Description	Rule Type	State	Rule Grouping	Rule Ordering	Selection
TTD	Term To Date	Term To Date	Active			

Rule Line

Line	*Line Granularity	*Period Type	*Start Date	*Date Determinant	*Day of Month	*Period	*Value Type	*Result Date
1	Contract ID	Monthly/Day of Period	Start Date	End of Period	BLANK/25	Period To	Pattern Key Figure Type	Result Date

Key Configuration Description

Field	Meaning	Notes
Line	Identifier of the line	Must be unique
Line Granularity	Determines the size of each partition of data	Mandatory input that has to be inserted in the *Granularity* section of the *Signature* tab
Period Type	Determines the final pattern structure (for example, monthly or quarterly)	Mandatory input
Start Date	Starting date to compute the day count	Mandatory input
Date Determinant	Specifies how the date is determined	Mandatory input. Codes: • 1 = Start of period • 2 = Middle of period • 3 = End of period • 4 = Actual day of period • 5 = Day of period **i Note** If you use code "5", the *Day of Month* field is mandatory.
Day of Month	Determines the result date if *Date Determinant* is set to *Day of Month*	
Period To	Day count of the starting pattern for the cash flow	Mandatory input
Value Type	Can have either the value "MOV" or "BAL", and determines which value type the cash flow item is comprised of.	Ensure that the field contains these two values only. This is important because different logic is applied for movements / balance types on the first date of the resulting pattern.
Result Date	The function writes the new date to this field.	Mandatory output

Expected Output

Expected Output for Input MOV (Delta Values) – Period Type = Monthly

Start Date	Contract ID	Pattern Key Figure Type	Period To	Period Type	*Result Date
2019-01-01	A	MOV (Delta)	1	6	2019-01-31

Start Date	Contract ID	Pattern Key Figure Type	Period To	Period Type	*Result Date
2019-01-01	A	MOV (Delta)	2	6	2019-02-28

Start Date	Contract ID	Pattern Key Figure Type	Period To	Period Type	*Result Date
2019-01-01	B	MOV (Delta)	1	6	2019-06-30
2019-01-01	B	MOV (Delta)	2	6	2019-07-31

Start Date	Contract ID	Pattern Key Figure Type	Period To	Period Type	*Result Date
2019-01-01	C	MOV (Delta)	1	6	2019-01-31
2019-01-01	C	MOV (Delta)	2	6	2019-02-28
2019-01-01	C	MOV (Delta)	3	6	2019-03-31
2019-01-01	C	MOV (Delta)	4	6	2019-04-30

Expected Output for Input MOV (Delta Values) – Period Type = Day of Period

Start Date	Contract ID	Pattern Key Figure Type	Period To	Period Type	*Result Date
2019-01-01	A	MOV (Delta)	1	6	2019-01-25
2019-01-01	A	MOV (Delta)	2	6	2019-02-25

Start Date	Contract ID	Pattern Key Figure Type	Period To	Period Type	*Result Date
2019-01-01	B	MOV (Delta)	1	6	2019-06-25
2019-01-01	B	MOV (Delta)	2	6	2019-07-25

Start Date	Contract ID	Pattern Key Figure Type	Period To	Period Type	*Result Date
2019-01-01	C	MOV (Delta)	1	6	2019-06-25
2019-01-01	C	MOV (Delta)	2	6	2019-07-25
2019-01-01	C	MOV (Delta)	3	6	2019-03-25
2019-01-01	C	MOV (Delta)	4	6	2019-04-25

Input Data with Pattern BAL (Cumulative) Values

Start Date	Contract ID	Pattern Key Figure Type	Period Unit	Period To
2019-01-01	D	BAL (cumulative)	7	0
2019-01-01	D	BAL (cumulative)	7	1
2019-02-01	E	BAL (cumulative)	11	0
2019-02-01	E	BAL (cumulative)	11	1
2019-02-01	E	BAL (cumulative)	11	2
2019-02-01	E	BAL (cumulative)	11	3

Flow Modeling Configuration

Rule Line

Line	*Line Granularity	*Period Type	*Start Date	*Date Determinant	*Day of Month	*Period	*Value Type	*Result Date
1	Contract ID	Quarterly	Start Date	End of Period		Period To	Pattern Key Figure Type	Result Date

Expected Output for Input BAL (Cumulative Values) – Period Type = Quarterly

Start Date	Contract ID	Pattern Key Figure Type	Period Unit	Period To	Result Date
2019-01-01	D	BAL (cumulative)	7	0	2019-01-01
2019-01-01	D	BAL (cumulative)	7	1	2019-01-31
2019-02-01	E	BAL (cumulative)	11	0	2019-02-01
2019-02-01	E	BAL (cumulative)	11	1	2019-03-31
2019-02-01	E	BAL (cumulative)	11	2	2019-06-30
2019-02-01	E	BAL (cumulative)	11	3	2019-09-30

1.4.2.6.8 Example: Value Conversion

Comprises two different calculation methods that can be used either to sum up cash flow items over a given set of terms (running total), or to calculate the delta values between a given set of cash flow items (balance = cumulative values, movement = delta values).

This example shows one of the methods. Starting with MOV data (delta values), we produce BAL data (cumulative values) for the cash flows.

With the configuration presented in the example, only the cash flow items of key figure type "Movement" are converted into balance values. Pattern items, which are of the type balance , will not be converted.

Input Data

Start Date	Contract ID	Pattern Key Figure Type	Period To	Value
2019-01-01	A	MOV (Delta)	1	USD 90
2019-01-01	A	MOV (Delta)	2	USD 120
2019-01-01	B	MOV (Delta)	1	USD 90
2019-01-01	C	MOV (Delta)	1	USD 0
2019-01-01	C	MOV (Delta)	2	USD 120

Flow Modeling Configuration

Rules (Tab)

Rule	Description	Rule Type	State	Rule Grouping	Rule Ordering	Selection
VC	Value Conversion	Value Conversion	Active			

Rule Line

Line	*Line Granularity	*Conversion Type	*Value Type Target	*Value Type Field	*Period To	*Value	*Value Result Field
1	Contract ID	Mov to Bal	Movement	Pattern Key Figure Type	Period To	Value	Result Amount

Key Configuration Description

Field	Meaning	Notes
Line	Identifier of the line	Must be unique
Line Granularity	Determines the size of one partition of data.	Mandatory input that has to be inserted in the *Granularity* tab of the *Signature* section.
Conversion Type	Specifies the calculation method used.	Mandatory input. "Mov to Bal" computes the running total, whereas "Bal to Mov" calculates delta values.
Value Type Target	Determines which key figure types (movement or balance) are converted	Mandatory input
Value Type Field	Can have the value "MOV" or "BAL", and determines which value type the cash flow item is comprised of.	Mandatory input Ensure that the field contains these two values only.
Period To	Determines the structure of the pattern.	Mandatory input
Value Result Field	The system writes the resulting amount to this field.	Mandatory output

Expected Output

In this example, the rule computes the running total of all cash flow items that have the value "MOV" in the *Pattern Key Figure Type*column.

Start Date	Contract ID	Pattern KF Type	Period To	Value	Result Amount
2019-01-01	A	MOV	1	USD 90	USD 90
2019-01-01	A	MOV	2	USD 120	USD 210
2019-01-01	B	MOV	1	USD 90	USD 90
2019-01-01	C	MOV	1	USD 0	USD 0
2019-01-01	C	MOV	2	USD 120	USD 120

Input Data BAL

Start Date	Contract ID	Pattern KF Type	Period To	Value
2019-01-01	A	BAL	1	USD 90
2019-01-01	A	BAL	2	USD 120
2019-01-01	B	BAL	1	USD 90

Start Date	Contract ID	Pattern KF Type	Period To	Value
2019-01-01	C	BAL	1	USD 0
2019-01-01	C	BAL	2	USD 120

The Flow Modeling configuration is the same; however, the *Conversion Type* is now "Bal to Mov" and *Value Type Target* is now "Balance".

Output Data BAL

Start Date	Contract ID	Pattern KF Type	Period To	Value	Result Amount
2019-01-01	A	BAL	1	USD 90	USD 90
2019-01-01	A	BAL	2	USD 120	USD 30
2019-01-01	B	BAL	1	USD 90	USD 90
2019-01-01	C	BAL	1	USD 0	USD 0
2019-01-01	C	BAL	2	USD 120	USD 120

1.4.2.6.9 Example: Incremental Value Calculation

The *Incremental Value Calculation* configuration, based on a vector of dates with two patterns (factors) assigned, combines the two factors and computes a mutual pattern.

In the example, we use a set of dates and their two factor patterns for a single contract as basis. The function does the following:

- It derives a new structure for the dates from the date of the pattern (start date in the example).
- It calculates a mutual pattern. Using the wording of the example, the mutual pattern is computed to retrieve the sum of the mutual pattern equal to the incurred pattern per incurred date. Moreover, the sum of the mutual pattern for a due date is equal to the sum of the due pattern for that due date (shown in the example).

Input Data

Contract ID	Date	Incurred Pattern	Due Pattern
A	2017-01-31	0.44	0.09
A	2017-02-28	0.36	0.37
A	2017-03-31	0.14	0.00
A	2017-04-30	0.06	0.29
A	2017-05-31	0.00	0.25

Flow Modeling Configuration

Rules (Tab)

Rule	Description	Rule Type	State	Rule Grouping Fields	Rule Ordering Fields	Selection
IVC	Incremental Value Calculation	Incremental Value Calculation	Active			

Rule Line

*Granularity Fields	*Period	*First Value Field	*Second Value Field	*First Period Field	*Second Period Field	*Value
Contract ID	Start Date	Incurred Pattern	Due Pattern	Incurred Date	Due Date	Mutual Pattern

Key Configuration Description

Field	Meaning	Notes
Line	Identifier of the line	Must be unique.
Granularity Fields	Determines the size of one partition of data	Mandatory input that has to be inserted in the *Granularity* section of the *Signature* tab.
Period	Term structure of the pattern	Mandatory input
First Value Field	First pattern value	Mandatory input
Second Value Field	Second pattern value	Mandatory input
First Period Field	Main date of the new term structure	Mandatory output
Second Period Field	Secondary date of the new term structure	Mandatory output
Value	Resulting mutual pattern	Mandatory output

Expected Output and Explanation

The following table shows the expected output:

Contract ID	Incurred Date	Due Date	Mutual Pattern
A	2017-01-31	2017-01-31	0.09
A	2017-01-31	2017-02-28	0.18
A	2017-01-31	2017-03-31	0.00

Contract ID	Incurred Date	Due Date	Mutual Pattern
A	2017-01-31	2017-04-30	0.09
A	2017-01-31	2017-05-31	0.08
A	2017-02-28	2017-02-28	0.19
A	2017-02-28	2017-03-31	0.00
A	2017-02-28	2017-04-30	0.09
A	2017-02-28	2017-05-31	0.08
A	2017-03-31	2017-03-31	0.00
A	2017-03-31	2017-04-30	0.08
A	2017-03-31	2017-05-31	0.06
A	2017-04-30	2017-04-30	0.03
A	2017-04-30	2017-05-31	0.03
A	2017-05-31	2017-05-31	0.00

Interim results explained below will be calculated by the system in all datasets based on dates from 2017-01-31 until 2017-05-31.

The following table is the input table with 5 unique records.

Contract ID	Date	Incurred Pattern	Due Pattern	
A	2017-01-31	0.44	0.09	Record 1
A	2017-02-28	0.36	0.37	Record 2
A	2017-03-31	0.14	0.00	Record 3
A	2017-04-30	0.06	0.29	Record 4
A	2017-05-31	0.00	0.25	Record 5

A) Explaining Output Field: First Period Field – in this example in the “Incurred Date” column

First Period Field starts with the date of the record from the input data. It produces 5 unique sets of data records.

Contract ID	Incurred Date	
A	2017-01-31	Taken from date of record 1
A	2017-02-28	Taken from date of record 2
A	2017-03-31	Taken from date of record 3
A	2017-04-30	Taken from date of record 4
A	2017-05-31	Taken from date of record 5

B) Explaining Output Field: Second Period Field – in this example in the “Due Date” column

Second Period Field produces records based on dates taken from records 1 to 5, starting from the date registered in the *First Period Field* until the date of the last record. For example, for record 1, it starts with due date 2017-01-31 and continues until the date of record 5, which is 2017-05-31.

Contract ID	Incurred Date	Due Date
A	2017-01-31	2017-01-31
		2017-02-28
		2017-03-31
		2017-04-30
		2017-05-31
A	2017-02-28	2017-02-28
		2017-03-31
		2017-04-30
		2017-05-31
A	2017-03-31	2017-03-31
		2017-04-30
		2017-05-31
A	2017-04-30	2017-04-30
		2017-05-31
A	2017-05-31	2017-05-31

At this point the new term structure is created.

C) Explaining Output Field: Value – in this example “Mutual Pattern”

Based on the *Primary Period Field* (*Incurred Date*), the sum of the mutual pattern for that incurred date that is equal to the incurred pattern of that incurred date (it is indicated only for the first incurred date, but it is true also for the other dates).

The following table shows the pattern of the input data again:

Contract ID	Start Date	Incurred Pattern	Due Pattern	Record
A	**2017-01-31**	**0.44**	0.09	Record 1
A	2017-02-28	0.36	0.37	Record 2
A	2017-03-31	0.14	0.00	Record 3
A	2017-04-30	0.06	0.29	Record 4
A	2017-05-31	0.00	0.25	Record 5

Contract ID	Incurred Date	Due Date	Mutual Pattern
A	**2017-01-31**	2017-01-31	**0.09**

Contract ID	Incurred Date	Due Date	Mutual Pattern
A	**2017-01-31**	2017-02-28	**0.18**
A	**2017-01-31**	2017-03-31	**0.00**
A	**2017-01-31**	2017-04-30	**0.09**
A	**2017-01-31**	2017-05-31	**0.08**
A	2017-02-28	2017-02-28	0.19
A	2017-02-28	2017-03-31	0.00
A	2017-02-28	2017-04-30	0.09
A	2017-02-28	2017-05-31	0.08
A	2017-03-31	2017-03-31	0.00
A	2017-03-31	2017-04-30	0.08
A	2017-03-31	2017-05-31	0.06
A	2017-04-30	2017-04-30	0.03
A	2017-04-30	2017-05-31	0.03
A	2017-05-31	2017-05-31	0.00

The sum of the first five mutual patterns is 0.44. The same as the incurred pattern for the corresponding start date.

D) Explaining Output Field: Value – in this example "Mutual Pattern"

Based on the Secondary Period Field (*Due Date*), the sum of the mutual pattern for a due date is equal to the due pattern of that due date (it is indicated only for the second due date, but it is true also for the other dates).

Contract ID	Start Date	Incurred Pattern	Due Pattern	
A	2017-01-31	0.44	0.09	Record 1
A	**2017-02-28**	0.36	**0.37**	Record 2
A	2017-03-31	0.14	0.00	Record 3
A	2017-04-30	0.06	0.29	Record 4
A	2017-05-31	0.00	0.25	Record 5

Contract ID	Incurred Date	Due Date	Mutual Pattern
A	2017-01-31	2017-01-31	0.09
A	2017-01-31	**2017-02-28**	**0.18**
A	2017-01-31	2017-03-31	0.00
A	2017-01-31	2017-04-30	0.09
A	2017-01-31	2017-05-31	0.08
A	2017-02-28	**2017-02-28**	**0.19**
A	2017-02-28	2017-03-31	0.00

Contract ID	Incurred Date	Due Date	Mutual Pattern
A	2017-02-28	2017-04-30	0.09
A	2017-02-28	2017-05-31	0.08
A	2017-03-31	2017-03-31	0.00
A	2017-03-31	2017-04-30	0.08
A	2017-03-31	2017-05-31	0.06
A	2017-04-30	2017-04-30	0.03
A	2017-04-30	2017-05-31	0.03
A	2017-05-31	2017-05-31	0.00

In the same way as for the *Second Value Field* for record 2, the field *Due Pattern* is also 0.37.

1.4.2.6.10 Example: Redistribution

This function calculates estimate values before the Reference Date for Redistribution (RDR) and redistributes them to the future periods after this date.

There are two possible methods: *Snow Canon* (SC) and *Snow Plough* (SP).

In the Snow Plough method the values are redistributed to the first value date field after the RDR.

In the Snow Canon method the values are redistributed proportionally to all the value date fields after the RDR. The following two examples show the differences between the two methods.

Input Data for Snow Canon

Contract ID	Incurred Date	Reported Date	Due Date	Reference Date of Redistribu-tion	Redistribution Method	Amount
A	2017-01-01	2017-02-01	2017-03-01	2017-02-23	SC	200
A	2017-01-01	2017-02-01	2017-04-01	2017-02-23	SC	400
A	2017-01-01	2017-04-01	2017-05-01	2017-02-23	SC	300
A	2017-01-01	2017-04-01	2017-06-01	2017-02-23	SC	100

Flow Modeling Configuration

Rules (Tab)

Rule	Description	Rule Type	State	Rule Grouping	Rule Ordering	Selection
RED	Redistribution	Redistribution	Active			

Input Fields

*Redistribution Method	*Reference Date of Redistribution	*Granularity Fields	*Date Determinant	Day of Month
Redistribution Method	Redistribution Reference Date	Incurred Date	Start Day of Period	

Changing Fields

*Value Date Field	*Value	Cleanup Fields
Reported Date	Amount	Due Date

Key Configuration Description

Field	Meaning	Notes
Redistribution Method	Indicates whether the redistribution method is “Snow Canon” or “Snow Plough”.	Mandatory input. Values: • SC: Snow Canon • SP: Snow Plough
Reference Date for Redistribution	Represents the threshold for the redistribution of the amounts.	Mandatory input
Granularity Field	Determines the size on one partition of data.	Mandatory input that has to be added to the *Granularity* section in the *Signature* tab.
Date Determinant	Specifies how the date should be determined.	Mandatory input Possible values: • Start date of the period • Mid date of the period • End date of the period • Actual date of period • Day of period
Day of Month	Indicates the day of the month where *Date Determinant* is set equal to *Day of Period*.	

Field	Meaning	Notes
Value Adjustment Granularity Field	Determines the granularity fields necessary for the value adjustment.	Mandatory input that has to be added to the *Granularity* section on the *Signature* tab.
Period Type	Determines the periodicity of the date in the *Value Date Field* (monthly, quarterly, yearly).	
Value Date Field	Date that is compared to *Reference Date for Redistribution*.	Mandatory input/output. Values in this field where the date is before the *Reference Date for Redistribution* will be redistributed (depending on the method).
Value	Amount for the redistribution	Mandatory input/output
Cleanup Field	This date is adjusted to perform the redistribution.	

Redistribution

Redistributed Records: The system selects the data with a *Reported Date* before the "Redistribution Reference Date".

Contract ID	Incurred Date	Reported Date	Due Date	Redistribution Reference Date	Redistribution Method	Amount
A	2017-01-01	2017-02-01	2017-03-01	2017-02-23	SC	200
A	2017-01-01	2017-02-01	2017-04-01	2017-02-23	SC	400

Records to be Redistributed with Allocation Factors: The system calculates the allocation factors to redistribute the amounts.

Contract ID	Incurred Date	Reported Date	Due Date	Redistribution Reference Date	Redistribution Method	Amount	
A	2017-01-01	2017-04-01	2017-05-01	2017-02-23	SC	300	75% (= 300/400)
A	2017-01-01	2017-04-01	2017-06-01	2017-02-23	SC	100	25% (=100/400)

The amount of the new record is calculated as follows:

- 200 x 0.75 = 150
- 200 x 0.25 = 50

- 400 x 0.75 = 300
- 400 x 0.25 = 100

Exected Result for Snow Canon

Contract ID	Incurred Date	Reported Date	Due Date	Reference Date of Redistribu-tion	Redistribution Method	Amount
A	2017-01-01	2017-04-01	2017-05-01	2017-02-23	SC	150
A	2017-01-01	2017-04-01	2017-05-01	2017-02-23	SC	300
A	2017-01-01	2017-04-01	2017-05-01	2017-02-23	SC	300
A	2017-01-01	2017-04-01	2017-06-01	2017-02-23	SC	50
A	2017-01-01	2017-04-01	2017-06-01	2017-02-23	SC	100
A	2017-01-01	2017-04-01	2017-06-01	2017-02-23	SC	100

Input Data for Snow Plough

Contract ID	Incurred Date	Reported Date	Due Date	Reference Date of Redistribu-tion	Redistribution Method	Amount
A	2017-01-01	2017-02-01	2017-03-01	2017-02-23	SP	200
A	2017-01-01	2017-02-01	2017-04-01	2017-02-23	SP	400
A	2017-01-01	2017-04-01	2017-05-01	2017-02-23	SP	300
A	2017-01-01	2017-04-01	2017-06-01	2017-02-23	SP	100
A	2017-01-01	2017-05-01	2017-06-01	2017-02-23	SP	500

The flow modeling configuration is the same as in the previous example. The only difference is the value in the field *Redistribution Method* ("SP" stands for the Snow Plough method).

Expected Output for Snow Plough

Contract ID	Incurred Date	Reported Date	Due Date	Reference Date of Redistribu-tion	Redistribution Method	Amount
A	2017-01-01	2017-04-01	2017-05-01	2017-02-23	SP	150
A	2017-01-01	2017-04-01	2017-05-01	2017-02-23	SP	300

Contract ID	Incurred Date	Reported Date	Due Date	Reference Date of Redistribu-tion	Redistribution Method	Amount
A	2017-01-01	2017-04-01	2017-05-01	2017-02-23	SP	300
A	2017-01-01	2017-04-01	2017-06-01	2017-02-23	SP	50
A	2017-01-01	2017-05-01	2017-06-01	2017-02-23	SP	0
A	2017-01-01	2017-04-01	2017-06-01	2017-02-23	SP	100
A	2017-01-01	2017-05-01	2017-06-01	2017-02-23	SP	0
A	2017-01-01	2017-04-01	2017-06-01	2017-02-23	SP	100
A	2017-01-01	2017-05-01	2017-06-01	2017-02-23	SP	500

1.4.2.6.11 Example: Scale Factor

Ratio of two corresponding values with similar field/data types (division).

In the example, we calculate the weighted claim (claim/total premium) for a pattern for three terms for a contract.

Input Data

Contract ID	Date	Total Premium	Claim
A	2019-01-01	1000	100
A	2019-02-01	1000	200
A	2019-03-01	1000	300

Flow Modeling Configuration

Rule	Description	Rule Type	State	Rule Grouping	Rule Ordering	Selection
SF	Scale Factor	Scale Factor	Active			

Line	*Numerator Value	*Denumerator Value	Scale Factor
1	Claim	Total Premium	Weighted Claim

Key Configuration Description

Field	Meaning	Notes
Line	Identifier of the line	Must be unique.
Numerator Value	Numerator of the ratio	Mandatory input
Denominator Value	Denominator of the ratio	Mandatory input
Scale Factor	Fields where the output is written	Mandatory output. The user has to add the field in the *Action* section of the *Signature* tab.

Expected Output

Contract ID	Date	Total Premium	Claim	Weighted Claim
A	2019-01-01	1000	100	0.1
A	2019-02-01	1000	200	0.2
A	2019-03-01	1000	300	0.3

1.4.2.6.12 Example: Scaling

Applies a scale factor through multiplication.

Input Data

Contract ID	Date	Weight	Ultimate
A	2019-01-01	0.33	1000
A	2019-02-01	0.33	1000
A	2019-03-01	0.33	1000

Flow Modeling Configuration

Rule	Description	Rule Type	State	Rule Grouping	Rule Ordering	Selection
SC	Scaling	Scaling	Active			

Line	*Value	*Scale Factor	Granularity Fields	*Scaled
1	Ultimate	Weight	Contract ID	Scale Value

Key Configuration Description

Field	Meaning	Notes
Line	Identifier of the line	Must be unique
Value	Multiplicand	Mandatory input
Scale Factor	Multiplier	Mandatory input
Granularity Fields	Determines the size on one partition of data	Mandatory input. The user has to add the field in the *Granularity* section of the *Signature* tab.
Scaled	Field where the output is written	Mandatory output. The user has to add the field in the *Action* section of the *Signature* tab.

Interim Result before Value Adjustment

Contract ID	Date	Weight	Ultimate	Scale Value
A	2019-01-01	0.33	1000	330
A	2019-02-01	0.33	1000	330
A	2019-03-01	0.33	1000	330

> **i** Note
>
> 1000 - 330 - 330 - 330 =10 added to last record.

Expected Data

Contract ID	Date	Weight	Ultimate	Scale Value
A	2019-01-01	0.33	1000	330
A	2019-02-01	0.33	1000	330
A	2019-03-01	0.33	1000	340

1.4.2.6.13 Example: Acknowledgment of Actuals

Enriches actuals by applying matching logic to determine date information missing earlier in the life cycle, and also determines the regime for every cashflow item.

The matching of actuals to estimates can be carried out in the following ways:

- If the CF calculation is “01”, the system matches the actual to the estimate based on the business date of the actual. This means, *Settled Date* for settled transactions, *Due Date* for due transactions and *Reported Date* for reported actuals.
- If the CF calculation is “02”, the system matches the actual to the estimates based only on the Secondary Risk Incurred Date. This is used in L&H business where the missing dates in the actuals are predetermined by a reverse life cycle conversion based on models.

In this example the CF calculation is “01”and the dates for an actual (Cf Indicator = 03) are enriched starting from a pattern of estimates (Cf Indicator = 01). The tables below contain the most important values and parameters.

CF_CALC

01	Used for P&C
02	Used for L&H

CF_INDICATOR

01	Estimate
02	Reported Actual
03	Due Business Transaction
04	Settled Business Transaction

REGIME

01	Follow Actuals
02	Reflect Actuals
03	Follow Estimates
04	Estimated Future

PERIOD TYPE

Monthly
Quarterly
Annual

Input Data

Cf Calc	Contract	Coverage	Cost Category	Cf Indicator	Pr Incurred Date	Pr Reported Date	Incurred Date	Reported Date	Due Date	Settled Date	Settled Amount	Currency	BT ID	Hold Back Date	RA HBD	Key Date	Regime
01	RIC_Q101DT	COV_Q101DT	1010	1	2018-01-01	2018-02-25	2018-01-25	2018-02-25	2018-03-25	2018-04-25	100	EUR		2018-01-31	2018-01-31	2018-05-15	
01	RIC_Q101DT	COV_Q101DT	1010	1	2018-01-01	2018-03-25	2018-01-25	2018-03-25	2018-04-25	2018-05-25	200	EUR		2018-01-31	2018-01-31	2018-05-15	
01	RIC_Q101DT	COV_Q101DT	1010	1	2018-01-01	2018-04-25	2018-03-25	2018-04-25	2018-05-25	2018-06-25	300	EUR		2018-01-31	2018-01-31	2018-05-15	
01	RIC_Q101DT	COV_Q101DT	1010	3					2018-05-14	2018-06-25	150	EUR	DUE_01	2018-01-31	2018-01-31	2018-05-15	
01	RIC_Q101DT	COV_Q101DT	1010	1	2018-01-01	2018-05-25	2018-04-25	2018-05-25	2018-06-25	2018-07-25	400	EUR		2018-01-31	2018-01-31	2018-05-15	

Flow Modeling Configuration

Rule	Description	Rule Type	State	Rule Grouping	Rule Ordering	Selection
AOA	Acknowledgement of Actuals Same Period	Acknowledge of Actuals	Active			

Input Fields

***Actuarial Granularity Fields**	**Contract, Cost Category, Coverage**
***BT Granularity Fields**	BT ID
***First Hold Back Date Field**	Hold Back Date
***Second Hold Back Date Field**	Ra HBD
***Business Date Field**	Key Date
***Period Type**	Monthly
***Life Cycle Step Field**	Cf Indicator
CF Calculation Field	CF Calc
Match Basis	Same Period Match

Changing Fields

*Amount Field	Settled Amount
Prim. Risk Incurred Date Field	Incurred Date
*Sec. Risk Incurred Date Field	Pr. Incurred Date
Prim. Risk Reported Date Field	Reported Date
Sec. Risk Reported Date Field	Pr. Reported Date
Due Date Field	Due Date
*Settled Date Field	Settled Date
Regime Field	Regime

Key Configuration Description

Field	Meaning	Notes
Actuarial Granularity Fields	List of fields that uniquely identify a set of cash flows as a group	Mandatory input. The fields have to be inserted into the *Granularity* section of the *Signature* tab.
BT Granularity Fields	List of fields that uniquely identify a set of cash flows as a group	Mandatory input
First Hold Back Date Field	Defines the date after which the model-based cashflows take precedence.	
Second Hold Back Date Field	Defines the date from which additional incurred date has to be calculated as a factor of actuals to be accounted as estimates.	
Business Date Field	Defines the key date for which the estimated cashflows are projected.	
Period Type	Defines the periodicity used as a basis for matching actuals to estimates.	
Life Cycle Step Field	Defines the definition of cashflow: estimate and various actual types.	Mandatory input
CF Calculation Field	Distinguishes between Life & Health and Property & Casualty Businesses.	
Match Basis	Determines the matching logic.	Possible Values: • Same period match • Past period match • Future period match • All periods match
Regime Field	Determines the regime to which the cashflow item belongs to.	

Output Result

Cf Calc	Contract	Coverage	Cost Category	Cf Indicator	Pr Incurred Date	Pr Reported Date	Incurred Date	Reported Date	Due Date	Settled Date	BT ID	Hold Back Date	Key Date	RA HBD	Regime	Settled Amount	Currency
01	RIC_Q101DT	COV_Q101DT	1010	03	2018-01-01	2018-04-25	2018-03-25	2018-04-25	2018-05-14	2018-06-25	DUE_01	2018-01-31	2018-05-15	2018-01-31	3	150	EUR
01	RIC_Q101DT	COV_Q101DT	1010	01	2018-01-01	2018-02-25	2018-01-25	2018-02-25	2018-03-25	2018-04-25		2018-01-31	2018-05-15	2018-01-31	3	100	EUR
01	RIC_Q101DT	COV_Q101DT	1010	01	2018-01-01	2018-03-25	2018-02-25	2018-03-25	2018-04-25	2018-05-25		2018-01-31	2018-05-15	2018-01-31	3	200	EUR
01	RIC_Q101DT	COV_Q101DT	1010	01	2018-01-01	2018-04-25	2018-03-25	2018-04-25	2018-05-14	2018-06-25		2018-01-31	2018-05-15	2018-01-31	3	300	EUR
01	RIC_Q101DT	COV_Q101DT	1010	01	2018-01-01	2018-05-25	2018-04-25	2018-05-25	2018-06-25	2018-07-25		2018-01-31	2018-05-15	2018-01-31	1	400	EUR

1.4.2.6.14 Example: Life Cycle Conversion

Applies the lags and lag factors delivered by the actuarial input to the cash flow stream in the form of a lag factor pattern, to determine the amounts and date for the lifecycle stage.

The example below shows a cash flow pattern. In this pattern, the reported date is one month after the incurred date. An amount is shown for each line of the pattern. What we want to do is to determine a new date (due date), based on the other dates, using the factor to spread the reported amount.

Input Data

Contract	Coverage	Cost Category	Cf Indicator	Incurred Date	Reported Date	Due Date	Cal Freq Code	Reported Amount	Currency	Factor
RIC_Q101DT	COV_Q101DT	1010	01	2019-02-25	2019-03-25		6	200	EUR	0.4
RIC_Q101DT	COV_Q101DT	1010	01	2019-02-25	2019-03-25		6	200	EUR	0.6

Contract	Coverage	Cost Category	Cf Indicator	Incurred Date	Reported Date	Due Date	Cal Freq Code	Reported Amount	Currency	Factor
RIC_Q101DT	COV_Q101DT	1010	01	2019-03-25	2019-04-25		6	300	EUR	0.4
RIC_Q101DT	COV_Q101DT	1010	01	2019-03-25	2019-04-25		6	300	EUR	0.6
RIC_Q101DT	COV_Q101DT	1010	01	2019-04-25	2019-05-25		6	400	EUR	0.4
RIC_Q101DT	COV_Q101DT	1010	01	2019-04-25	2019-05-25		6	400	EUR	0.6

Flow Modeling Configuration

Rules

Rule	Description	Rule Type	State	Rule Grouping	Rule Ordering	Selection
LFC	R2D	Life Cycle Conversion	Active			

Input Fields

Life Cycle Reversed	**Life Cycle Conversion**
Client Reporting Frequency	
Date Determinant	Day of Period
Date Field	Reported Date
Date of Month	25
Lag Factor Value	Factor
Lag Factor Frequency	Cal Freq Code
LCC Frequency Field	Contract, Cost Category, Coverage, Incurred Date, Reported Date
Period	Period
Value	Reported Amount

Output Fields

Life Cycle Reversed	**Life Cycle Conversion**
Life Cycled Amount	Due Amount
Life Cycled Date	Due Date

Key Configuration Description

Fields	Meaning	Notes
Life Cycle Reversed	Defines whether the life cycle process determines a date in the future or in the past.	Mandatory input
Client Reporting Frequency	Only applies in the incurred to reported life cycle convention step.	The value is delivered as part of the master data.
Date Determinant	Defines how the date should be determined.	Mandatory input. Codes: • 1, Start of the period • 2, Mid of the period • 3, End of the period • 4, Actual day of the period
Date Field	Defines the date which will be used as the input for the Life Cycle Conversion Step.	Mandatory input
Date of Month	Defines the day of the result date according to the configuration.	
Lag Factor Value	Defines the factors to be applied for determining the amounts per life cycled date.	Mandatory input
Lag Factor Frequency	Defines the periodicity of the lag factors (monthly, quarterly, annual etc...)	Mandatory input
LCC Granularity Field	A list of fields that uniquely identify a set of cash flows as a group.	Mandatory input
Period	A list of fields that uniquely identify a set of cash flows as a group.	Mandatory input
Value	Defines the number of periods to be applied for the date determination.	Mandatory input
Life Cycled Amount	Field where the resulting amount is written.	Mandatory output
Life Cycled Date	Field where the resulting date is written.	Mandatory output

Expected Output

Contract	Coverage	Cost Category	Cf Indicator	Incurred Date	Reported Date	Due Date	Reported Amount	Period	Factor	Due Amount	Currency	Cal Freq Code
RIC_Q101DT	COV_Q101DT	1010	01	2019-02-25	2019-03-25	2019-03-25	200	000000	0.4	80	EUR	6
RIC_Q101DT	COV_Q101DT	1010	01	2019-02-25	2019-03-25	2019-04-25	200	000001	0.6	120	EUR	6
RIC_Q101DT	COV_Q101DT	1010	01	2019-03-25	2019-04-25	2019-04-25	300	000000	0.4	120	EUR	6
RIC_Q101DT	COV_Q101DT	1010	01	2019-03-25	2019-04-25	2019-05-25	300	000001	0.6	180	EUR	6
RIC_Q101DT	COV_Q101DT	1010	01	2019-04-25	2019-05-25	2019-05-25	400	000000	0.4	160	EUR	6
RIC_Q101DT	COV_Q101DT	1010	01	2019-04-25	2019-05-25	2019-06-25	400	000001	0.6	240	EUR	6

1.4.2.6.15 Example: Modulation In

The table below explains the general concept of Modulation In.

Scenarios	Previous Contract	Following Contract	Rule
First scenario	No	Yes	ModIn totals ModOut multiplied by –1 with *Incurred Date* = "Contract Start Date"
Second scenario	No	Yes	ModIn totals ModOut multiplied by –1 with *Incurred Date* = "Contract Start Date"
			ModIn of following contract multiplied by –1 with *Incurred Date* = "Start Date of following Contract"
Third scenario	Yes	Yes	ModIn should be calculated
			ModIn of following contract multiplied by –1 with *Incurred Date* = "Start Date of following Contract"

The following example shows how this configuration is calculated in practice.

Input Data

Contract	Coverage	Old Quota Share R.	Quota Share R.	Value Mod
CONTRACT_A	COVERAGE_A	10	10	100
CONTRACT_A	COVERAGE_B	10	20	100
CONTRACT_A	COVERAGE_C	10	5	100
CONTRACT_B	COVERAGE_A	10	10	100
CONTRACT_B	COVERAGE_B	10	20	200
CONTRACT_B	COVERAGE_C	0.1	1	1

Flow Modeling Configuration

Rules

Rule	Description	Rule Type	State	Rule Grouping Fields	Rule Ordering Fields	Selection
MDI	Modulation In	Modulation In	Active			

Input Fields

Old Share Field	Contract End Date
New Share Field	Exposure Date
Granularity Field	Contract Coverage

Changing Fields

Value	Value Mod

Calculation logic: (New Share Field/Old Share Field)*(–Value)

Expected Output

Contract	Coverage	Old Quota Share R.	Quota Share R.	Value Mod
CONTRACT_A	COVERAGE_A	10	10	–100
CONTRACT_A	COVERAGE_B	10	20	–200
CONTRACT_A	COVERAGE_C	10	5	–50
CONTRACT_B	COVERAGE_A	10	10	–100
CONTRACT_B	COVERAGE_B	10	20	–400
CONTRACT_B	COVERAGE_C	0.1	1	–10

1.4.2.6.16 Example: Modulation Out

Selects amounts in the basis cash flow where the exposure date is greater than the coverage end date. If Modulation Out for contract T is updated and contract T+1 exists, Modulation In for contract T+1 must be (re-)calculated.

Scenarios	Previous Contract	Following Contract	Rule
First scenario	No	No	ModOut is based on UnMod
Second scenario	No	Yes	ModOut is based on UnMod
Third scenario	Yes	Yes	ModOut is based on UnMod

Here is an example to explain the computation that this configuration does in practice.

Input Data

Contract	Contract End Date	Exposure Date	Value Mod
CONTRACT_D	2019-01-01	2019-01-01	100
CONTRACT_A	2019-01-01	2019-01-19	200
CONTRACT_B	2019-01-01	2019-01-15	300
CONTRACT_C	2019-01-01	2019-01-02	400

If the exposure date is greater than the contract end date, the system calculates the days between the two dates, calculates a factor based on the day counts (depending on the *Day Count Convention*) and multiplies the value for that factor with a negative sign.

For the second line of the input data (the first has Contract End Date=Exposure Date and is excluded from the computation) we will get:

- 18 days between the two dates
- 30 days for January (in the convention German 30/360 all the months have 30 days).

So, the result will be (18/30)*(–200)= –120

Flow Modeling Configuration

Rules (Tab)

Rule	Description	Rule Type	State	Rule Grouping Fields	Rule Ordering Fields	Selection
MDO	Modulation	Modulation Out	Active			

Input Fields

Contract End Date	Contract End Date
Exposure Date	Exposure Date
Day Count Convention	German 30/360

Changing Fields

Value	Value Mod

Key Configuration Description

Field	Meaning	Notes
Contract End Date	End date of the contract	Mandatory input
Exposure Date	Exposure date	Mandatory input. If Exposure Date > Contract End Date, Modulation Out is performed.
Day Count Convention	Convention to compute the day count	Mandatory input
Value	Value field	Mandatory output

Expected Output

Contract	Contract End Date	Exposure Date	Value Mod
CONTRACT_A	2019-01-01	2019-01-19	–120
CONTRACT_B	2019-01-01	2019-01-15	–140
CONTRACT_C	2019-01-01	2019-01-02	–13.33

1.4.2.6.17 Example: Item Number Generation

Separates each partition by creating a number for each one. To do this, the system needs a *Granularity* field (which separates the partitions from each other) and an *Item Number* field (which is filled by this rule type).

In the example, we mark the cash flow pattern according to the version number: when the version number increases, we add one to the *Item Number*. Only the *Granularity* field (Contract ID) is relevant for the calculation.

Input Data

Version ID	Start Date	Contract ID	Period From	Period To
1	2019-01-01	A	1	3
1	2019-01-01	A	4	4
2	2020-01-16	B	1	2
2	2020-01-16	B	3	8
3	2019-01-01	C	1	3
4	2019-01-01	D	1	2
4	2019-01-01	D	3	8

Flow Modeling Configuration

Rules

Rule	Description	Rule Type	State	Rule Grouping Fields	Rule Ordering Fields	Selection
IN	CF Item Number	Item Number Generation	Active			

Rule Line

Line	*Item Number Granularity Fields	*Item Number
L1	Contract ID	Item ID

Key Configuration Description

Field	Meaning	Notes
Line	Identifier of the line	Must be unique. A single Key Figure Formula Rule can run more lines.
Item Number Granularity Field	Determines the size of one partition of data	Mandatory input. You have to add the field in the *Granularity* section of the *Signature* tab.
Item Number	Fields where the output is written	Mandatory output. You have to add the field in the *Action* section of the *Signature* tab.

Expected Output

Version ID	Start Date	Contract ID	Item ID
1	2019-01-01	A	1
1	2019-01-01	A	2
2	2020-01-16	B	1
2	2020-01-16	B	2
3	2019-01-01	C	1
4	2019-01-01	D	1
4	2019-01-01	D	2

1.4.2.6.18 Example: Cash Flow Regime

This configuration is helpful to manage cash flows that follow the distinction between Estimates, Reported Actuals, Due Business Transaction and Settled Business Transaction.

This function adjusts the cashflow stream based on the regime into which each of the cashflows falls. In the context of ECP the regimes that are managed are:

- Follow Actuals (01)
  This regime comprises only the effect of actuals. Therefore the system removes any estimate item that falls in this regime from the final cashflow.
- Reflect Actuals (02)
  This regime comprises only the effect of actuals. Model-based estimates do not have any effect in this regime. However, more actuals may be expected to be reported in this period. The system therefore calculates an additional incurred estimate as a factor of the actuals. These additional incurred estimates will have the same date information as that of the actuals but the amounts will be calculated as a factor of the actuals and will applythe formula (Amount * Factor / (1 - Factor)).
- Follow Estimates (03)
  In this regime, the model-based estimates are expected to be effective. Therefore, for every actual that falls into this regime an additional negated estimate is introduced into the cashflow with the same date information as the actuals.
- Estimated Future (04)
  In this regime, no actual information is expected.

We show three simple examples for each of the three regimes (the first three require configuration because different behaviors are expected). The key fields and parameters of the function are explained below.

Values for Key Fields of the Examples

CF_CAL

01	P&C
02	L&H

CF_INDICATOR

01	Estimate
02	Reported Actual
03	Due Business Transaction
04	Settled Business Transaction

REGIME

01	Follow Actuals
02	Reflect Actuals
03	Follow Estimate
04	Estimated Future

Example: Follow Actuals

Input Data

Cf Calc	Contract	Coverage	Cost Category	Cf Indicator	Pr Incurred Date	Pr Reported Date	Incurred Date	Reported Date	Due Date	Settled Date	Settled Amount	Currency	BT_ID	Hold Back Date	RA HBD	Key Date	Regime	Factor
02	RIC_Q101DT	COV_Q101DT	1010	01	2017-11-01	2017-12-25	2017-11-25	2017-12-25	2018-01-25	2018-02-25	50	EUR		2018-01-31	2017-11-30	2018-05-15	01	0.000
02	RIC_Q101DT	COV_Q101DT	1010	03	2018-01-01	2018-04-25	2018-03-25	2018-04-25	2018-05-14	2018-06-25	150	EUR	DUE_01	2018-01-31	2017-11-30	2018-05-15	01	0.000

Flow Modeling Configuration

Rules (the same for all regimes)

Rule	Description	Rule Type	State	Rule Grouping	Rule Ordering	Selection
CFR	CF Regime	Cash Flow Regime	Active			

Rule Line

Line	Line Type	Description
L001	**CF Regime: Follow Actuals**	FA

*Input Fields

Life Cyle Step Field	Cf Indicator
Regime	**Regime**

Key Configuration Description

Field	Meaning	Notes
Life Cycle Step Field	Indicates the nature of the entry	Mandatory input
Regime	Indicates the regime that has to be applied	Mandatory input

Expected Output

Cf Calc	Contract	Coverage	Cost Category	Cf Indicator	Pr Incurred Date	Pr Reported Date	Incurred Date	Reported Date	Due Date	Settled Date	Settled Amount	Currency	BT ID	Hold Back Date	RA HBD	Key Date	Regime	Factor
02	RIC_Q101DT	COV_Q101DT	1010	03	2018-01-01	2018-04-25	2018-03-25	2018-04-25	2018-05-14	2018-06-25	150.000	EUR	DUE_01	2018-01-31	2017-11-30	2018-05-15	01	0.000

Example: Reflect Actuals

Input Data

Cf Calc	Contract	Coverage	Cost Category	Cf Indicator	Pr Incurred Date	Pr Reported Date	Incurred Date	Reported Date	Due Date	Settled Date	Settled Amount	Currency	BT ID	Hold Back Date	RA HDB	Key Date	Regime	Factor
02	RIC_Q101DT	COV_Q101DT	1010	03	2018-01-01	2018-02-25	2018-01-25	2018-02-25	2018-03-15	2018-04-25	75.000	EUR	DUE_02	2018-01-31	2017-11-30	2018-05-15	02	0.700
02	RIC_Q101DT	COV_Q101DT	1010	03	2018-01-01	2018-02-25	2018-01-25	2018-02-25	2018-03-15	2018-04-25	100.000	EUR		2018-01-31	2017-11-30	2018-05-15	02	0.700

Rule Line

Line	Line Type	Description
L001	**CF Regime: Reflect Actuals**	FA

*Input Fields

BT Granularity Fields	BT_ID
Amount	**Settled Amount**
Factor	**Factor**
Life Cycle Step Field	**Cf Indicator**
Regime	**Regime**

Key Configuration Description

Field	Meaning	Notes
BT Granularity Fields	Fields that define the partition of data	Mandatory input
Amount	Field that contains the amounts	Mandatory input
Factor	Factor that has to be applied	Mandatory input
Life Cycle Step Field	Indicates the nature of the entry	Mandatory input
Regime	Indicates the regime that has to be applied	Mandatory input

Expected Output

Cf Calc	Contract	Coverage	Cost Category	Cf Indicator	Pr Incurred	Pr Reported Date	Incurred Date	Reported Date	Due Date	Settled Date	BT ID	Currency	Factor	Hold Back Date	Key Date	RA HDB	Regime	Settled Amount
02	RIC_Q101DT	COV_Q101DT	1010	03	2018-01-01	2018-02-25	2018-01-25	2018-02-25	2018-03-15	2018-04-25	DUE_02	EUR	0.700	2018-01-31	2018-05-15	2017-11-30	02	75.000
02	RIC_Q101DT	COV_Q101DT	1010	01	2018-01-01	2018-02-25	2018-01-25	2018-02-25	2018-03-25	2018-04-25		EUR	0.700	2018-01-31	2018-05-15	2017-11-30	02	100.000
02	RIC_Q101DT	COV_Q101DT	1010	01	2018-01-01	2018-02-25	2018-01-25	2018-02-25	2018-03-15	2018-04-25		EUR	0.700	2018-01-31	2018-05-15	2017-11-30	02	175.000

The added amount 175 is computed according to the formula (Amount*Factor/(1-Factor)) as (75*0.7/(1-0.7)).

Example: Follow Estimates

Input Data

Cf Calc	Contract	Coverage	Cost Category	Cf Indicator	Pr Incurred Date	Pr Reported Date	Incurred Date	Reported Date	Due Date	Settled Date	Settled Amount	Currency	BT ID	Hold Back Date	RA HDB	Key Date	Regime	Factor
02	RIC_Q101DT	COV_Q101DT	1010	03	2018-01-01	2018-02-25	2018-01-25	2018-02-25	2018-03-15	2018-04-25	75.000	EUR	DUE_02	2018-01-31	2017-11-30	2018-05-15	03	0.700
02	RIC_Q101DT	COV_Q101DT	1010	01	2018-01-01	2018-02-25	2018-01-25	2018-02-25	2018-03-15	2018-04-25	100.000	EUR		2018-01-31	2017-11-30	2018-05-15	03	0.700

Rule Line

Line	Line Type	Description
L001	**CF Regime: Follow Estimates**	FA

*Input Fields

BT Granularity Fields	BT_ID
Amount	Settled Amount
Life Cycle Step Field	Cf Indicator
Regime	Regime

Key Configuration Description

Field	Meaning	Notes
BT Granularity Fields	Fields that define the partition of data	Mandatory input
Amount	Field that contains the amounts	Mandatory input
Life Cycle Step Field	Indicates the nature of the entry	Mandatory input
Regime	Indicates the regime that has to be applied	Mandatory input

Expected Output

Cf Calc	Contract	Coverage	Cost Category	Cf Indicator	Pr Incurred	Pr Reported Date	Incurred Date	Reported Date	Due Date	BT ID	Currency	Factor	Hold Back Date	Key Date	RA HDB	Regime	Settled Amount	Settled Date
02	RIC_Q101DT	COV_Q101DT	1010	03	2018-01-01	2018-02-25	2018-01-25	2018-02-25	2018-03-15		EUR	0.700	2018-01-31	2018-05-15	2017-11-30	03	–75.000	2018-04-25
02	RIC_Q101DT	COV_Q101DT	1010	01	2018-01-01	2018-02-25	2018-01-25	2018-02-25	2018-03-15	DUE_02	EUR	0.700	2018-01-31	2018-05-15	2017-11-30	03	75.000	2018-04-25
02	RIC_Q101DT	COV_Q101DT	1010	01	2018-01-01	2018-02-25	2018-01-25	2018-02-25	2018-03-25		EUR	0.700	2018-01-31	2018-05-15	2017-11-30	03	100.000	2018-04-25

The function added a new entry with a negative amount.

1.4.2.6.19 Example: Clear Actual Dates

Clears the actual date information in the cashflows arising from actuals.

We show below a simplified pattern. The function clears the dates according to a predefined logic that depends on the values in the field inserted in *Life Cycle Step* field. The result is the same pattern but with the dates cleared.

Input Data

CF Indicator	Contract	Coverage	Cost Category	PR Incurred Date	PR Reported Date	Incurred Date	Reported Date	Due Date	Settled Date	Settled Amount
03	RIC_Q101DT	COV_Q101DT	1010	2017-01-11	2017-12-25	2017-11-25	2017-12-25	2018-01-25	2018-02-25	50.000
01	RIC_Q102DT	COV_Q101DT	1010	2017-01-11	2017-12-25	2017-11-25	2017-12-25	2018-01-25	2018-02-25	50.000
02	RIC_Q103DT	COV_Q101DT	1010	2017-01-11	2017-12-25	2017-11-25	2017-12-25	2018-01-25	2018-02-25	50.000

Flow Modeling Configuration

Rules

Rule	Description	Rule Type	State	Rule Grouping	Rule Ordering	Selection
CAD	Clear Actual Dates	Clear Actual Dates	Active			

Rule Line

Line	*Life Cycle Step Field	*Pr. Risk Incurred Date	*Sec. Risk Incurred Date	*Pr. Risk Reported Date	*Sec. Risk Reported Date	*Due Date	*Settled Date
L1	CF Indicator	Pr. Incurred Date	Incurred Date	Pr. Reported Date	Reported Date	Due Date	Settled Date

Key Configuration Description

Field	Meaning	Notes
Line	Identifier of the line	Must be unique. A single Key Figure Formula Rule can run multiple lines.
Life Cycle Step Field	Defines the logic for cancellation of the dates	Mandatory input. Four codes are possible: • 1: The dates are not deleted • 2: All the dates are deleted except *Due Date* and *Settled Date* • 3: All the dates are deleted but *Settled Date* • 4: All the dates are deleted.
Pr. Risk Incurred	Date of the pattern	Mandatory input and output
Sec. Risk Incurred Date	Date of the pattern	Mandatory input and output
Pr. Risk Reported Date	Date of the pattern	Mandatory input and output
Sec. Risk Reported Date	Date of the pattern	Mandatory input and output
Due Date	Date of the pattern	Mandatory input and output
Settled Date	Date of the pattern	Mandatory input and output

Expected Output

Cf Indicator	Contract	Coverage	Cost Category	Pr Incurred Date	Pr Reported Date	Incurred Date	Reported Date	Due Date	Settled Date
01	RIC_Q101DT	COV_Q101DT	1010	2017-01-11	2017-12-25	2017-11-25	2017-12-25	2018-01-25	2018-02-25
02	RIC_Q101DT	COV_Q101DT	1010					2018-01-25	2018-02-25
03	RIC_Q101DT	COV_Q101DT	1010						2018-02-25
04	RIC_Q101DT	COV_Q101DT	1010						

1.4.2.6.20 Example: Incurred to Reported Factor Calculation

Calculates the *Incurred to Reported* lags in cases where these lags are not delivered directly, using the more granular *Policy Holder to Primary Insurer* and *Primary Insurer to Reinsurer* lags.

In brief, depending on the selections, the system computes a combined factor and a new structure for the cash flows.

Input

Input Data

Conctract	Coverage	Cost Category	RAAP	LFP Type	LFP SUB-CAT	Period	Cal Freq Code	Factor
RIC_Q101DT	COV_Q101DT	1010	000001	IN2RE	PH2PI	000000	11	0.200
RIC_Q101DT	COV_Q101DT	1010	000001	IN2RE	PH2PI	000001	11	0.100
RIC_Q101DT	COV_Q101DT	1010	000001	IN2RE	PH2PI	000002	11	0.700
RIC_Q101DT	COV_Q101DT	1010	000001	IN2RE	PI2RI	000001	11	0.600
RIC_Q101DT	COV_Q101DT	1010	000001	IN2RE	PI2RI	000002	11	0.400

Flow Modeling Configuration

Rules

Rule	Description	Rule Type	State	Rule Grouping	Rule Ordering	Selection
I2R	Incurred to Reported Factor Calculation	Incurred to Reported Factor Calculation	Active			`LFP_SUBCAT=(LFP_SUBCAT!='')`

Input Fields

Input Fields	Value
***Lag Factor Pattern Granularity Fields**	Cal Freq Code, Contract, Cost Category, Coverage, LFP Subcat, LFP Type
***Lag Factor Type Field**	LFP Type
***LFP Sub Category**	LFP Subcat
Insured to Insurer Lag Factor Type	Policy Holder to Primary Insurer
Insurer to Reinsurer Lag Factor Type	Primary Insurer to Reinsurer
***Lag Factor Type**	Incurred to Reported

Changing Fields

Changing Fields	Value
***Lag Factor Value Field**	Factor
***Period**	Period

Key Configuration Description

Field	Meaning	Notes
*Lag Factor Pattern Granularity Fields	Determines the size of one partition of data.	Mandatory input. You have to add the field in the *Granularity* section of the *Signature* tab.
*Lag Factor Type Field	Indicates the life cycle of the cash flow.	Mandatory input. The field must contains the following values: • IN2RE (Incurred to Reported) • RE2DU (Reported to Due) • DU2SE (Due to Settled)

Field	Meaning	Notes
*LFP Sub Category	Indicates the subcategory of the pattern.	Mandatory input. The field must contain the following values: • PH2PI (Policy Holder to Primary Insurer) • PI2RI (Primary Insurance to Reinsurer)
Insured to Insurer Lag Factor Type	Manages how to combine the data for the calculation (see the example).	
Insurer to Reinsurer Lag Factor Type	Manages how to combine the data for the calculation (see the example).	
*Lag Factor Type	Acts as a filter on the field *Lag Factor Type Field*	Mandatory input
*Lag Factor Value Field	Where the combined factor is written.	Mandatory input/output. You have to add the field in the *Action* section of the *Signature* tab.
*Period	Where the new period structure is written.	Mandatory input/output. You have to add the field in the *Action* section of the *Signature* tab.

Expected Output

Contract	Coverage	Cost Category	RAAP	LFP Type	LFP SUBCAT	Period	Cal Freq Code	Factor
RIC_Q101DT	COV_Q101DT	1010	000001	IN2RE	PH2PI	1	11	0.120
RIC_Q101DT	COV_Q101DT	1010	000001	IN2RE	PH2PI	2	11	0.140
RIC_Q101DT	COV_Q101DT	1010	000001	IN2RE	PH2PI	3	11	0.460
RIC_Q101DT	COV_Q101DT	1010	000001	IN2RE	PH2PI	4	11	0.280

Additional Information on Calculation Logic

PERIOD						
LFP_SUBCAT	000000	000001	000002	000003	000004	
PH2PI	0.2	0.1	0.7			
PI2RI	0	0.6	0.4			
Calculation for PH2PI = 0.2	0.2*0=0	0.2*0=0	0.2*0.4=0.08			
Calculation for PH2PI = 0.1		0.1*0=0	0.1*0.6=0.06	0.1*0.4=0.04		
Calculation for PH2PI = 0.7			0.7*0=0	0.7*0.6=0.42	0.7*0.0.4=0.28	
IN2RE	0	0.12+0=0.12	0.08+0.06+0=0.14	0.04+0.42=0.46	0.28	New Period Structure

1.4.2.6.21 Example: Factors for Additional Incurred

In the context of the Estimated Cash Flow Preparation, this function is usually used to calculate the factors to be applied in the *Reflect Actual Regime* based on the delivered *Policy Holder to Primary Insurer* lags.

Factors are calculated as a difference between 1 and the cumulated sum of the policy holder to primary insurer lag factors for a certain granularity. The number of periods is determined by the *Reflect Actuals* attachment point.

Input

Input Data

Contract	Coverage	Cost Category	RAAP	LFP Type	LFP SUB-CAT	Period	Cal Freq Code	Factor
RIC_Q101DT	COV_Q101DT	1010	000001	IN2RE	PH2PI	000000	11	0.200
RIC_Q101DT	COV_Q101DT	1010	000001	IN2RE	PH2PI	000001	11	0.100
RIC_Q101DT	COV_Q101DT	1010	000001	IN2RE	PH2PI	000002	11	0.700
RIC_Q101DT	COV_Q101DT	1010	000001	IN2RE	PI2RI	000001	11	0.600
RIC_Q101DT	COV_Q101DT	1010	000001	IN2RE	PI2RI	000002	11	0.400

Input Fields

Lag Factor Pattern Granularity Fields	Contract, Cost Category, Coverage
Lag Factor Frequency Field	Cal Freq Code
Date Determinant	Actual Date of Period
Period Field	Period
RA Attachment Pt. Field	RAAP
Hold Back Date Field	Hold Back Date

Flow Modeling Configuration

Rules (Tab)

Rule	Description	Rule Type	State	Rule Grouping	Rule Ordering	Selection
I2R	Factors for Additional Incurred Cash Flows	Factors for Additional Incurred Cash Flows	Active			`LFP_TYPE = 'IN2RE'; LFP_SUBCAT = 'PH2PI'`

Output

Output Fields

Lag Factor Pattern Granularity Fields	Factor
Lag Factor Frequency Field	Period Start Date
Date Determinant	Period End Date

Output Data

Contract	Coverage	Cost Category	RAAP	LFP Type	LFP SUBCAT	Period	Cal Freq Code	Factor	Hold Back Date	Period End Date	Period Start Date
RIC_Q101DT	COV_Q101DT	1010	000000	IN2RE	PH2PI	1	11	0.70	2018-01-31	2018-01-31	2018-01-01
RIC_Q101DT	COV_Q101DT	1010	000000	IN2RE	PH2PI	2	11	0.00	2018-01-31	2017-12-31	2017-12-01

1.4.2.7 Funds Transfer Pricing

The funds transfer pricing (FTP) methodology determines the cost of funds associated with the lending and borrowing from a financial institution (for example, a bank) while considering liquidity, interest rate and currency risks.

This mechanism is therefore also a part of the internal process that sets interest rates on the retail and commercial products of a bank. FTP measures the performance of the bank's deposit-raising (borrowing) and funds-advancing (lending) business units by constructing and evaluating deposits and loans profitability indicators (for example, FTP rate, costs of funding, net interest margin (NIM), and net interest income (NII)). The bank's treasury departments usually determine the following different FTP rate types:

1. Rates charged for providing funds to loan-giving business units
2. Compensation rates for funds received from deposit-raising business units

To meet the regulatory and functional needs for a sound FTP system, financial institutions can implement the *Funds Transfer Pricing* tool integrated in SAP Profitability and Performance Management that enables you to do the following:

- Generate various cash flows on both fixed and variable rate instruments using *Flow Generation* and *Rate Modeling* rule types
- Construct FTP cost of funds yield curve using multiple interpolation and smoothening methods
- Calculate various durations on selected instruments
- Apply cash-flow-based FTP rate calculation methods (term-weighted matched maturity, NPV approach, and so on)
- Apply non-cash-flow-based FTP rate calculation methods (caterpillar or strip funding approach, weighted average rates, pool rate assignment, and so on)
- Use or construct other fund and liquidity transfer pricing approaches

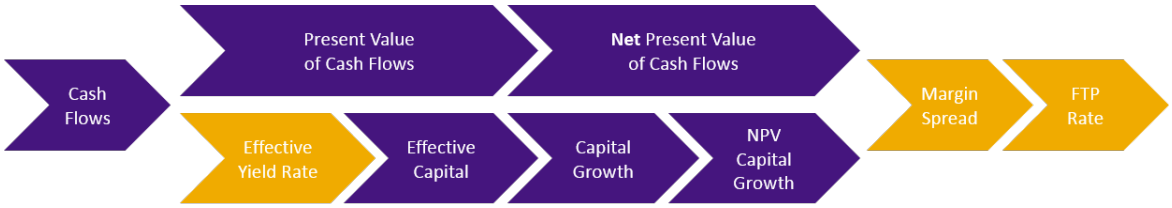
Various rule and line types of the FTP function enable the implementation of the intricate methodology outlined above.

Rule Types

The following rule types are available:

- **Durations**
  Calculates three different types of duration. Their outputs are of similar numerical value, but theoretical concepts behind calculations and the interpretation of results differ as do the practical applications.
- **Flow Generation**
  As an umbrella for six different line types, this rule type generates principal cash flows according to amortization pattern or flexible individual cash flow for disbursement, except principals by *Single Flow* line type. Note that interest cash flows are handled by the *Rate Modeling* rule type. Therefore, by combining different line types of *Flow Generation* and *Rate Modeling*, SAP Profitability and Performance Management is able to generate various types of future cash flows in real business scenarios.
- **Flow Merge**
  Joins financial position master data from one function with relevant transaction or business event data from another function that indicates exceptional cash flows.

- **Formula**
  Applies formulas and HANA SQL functions that return numeric and string values to output fields.
- **Market Interest Rate**
  Determines the FTP/LTP rate (the rate at which a bank extends or accepts loans to or from its internal departments) with several calculation steps in the logic sequence of the *Net Present Value Approach* shown below.



- **Matched Maturity**
  Generally, as an extension of the multiple pools FTP methodology, this approach involves the coordination of a financial institution's cash inflows with its cash outflows based on the maturities of its assets and liabilities. Specifically, it calculates the FTP rate by matching due dates of (asset & liability instruments') principal cash flows to the marginal cost of ffunds curve (also referred to as the FTP curve) rates of corresponding maturity.
- **Rate Modeling**
  As an umbrella for six different line types, this rule type covers various areas: it generates interest cash flows, interpolates rates on a yield curve, creates forward rates and determines daycount. Since the *Flow Generation* rule type generates principal cash flows in combination with *Rate Modeling*, SAP Profitability and Performance Management can generate various types of future cash flows in real business scenarios.
- **Running Total**
  This rule type sequentially sums a selected field and updates this sum (also referred to as the "running total") for each row by adding it to the previous running total. It is used in finance to calculate, for example, loan principal outstanding that is the basis for interest calculations.
- **Series Generation**
  This rule type generates series data by providing several parameters such as *Step Size*, *Series Type*, *Period From* and *Period To*.
- **Strip Funding**
  Divides the net present value of principal payments by the sum of the term-accrued period-end principal balances, while discounting both with matching funding rates to calculate the FTP rate.
- **Weighted Average Rate**
  This approach centers on the logic used to split a financial position into components. The chosen logic mandates matching of each part to different factors (weights) and funding rates – the only variables in the simple FTP weighted average rate calculation (WAR).

Rule Lines

Durations

The following rule lines are available:

- **Macaulay Duration** is the weighted average maturity of future cash flows of a financial instrument. The weight of each cash flow's maturity is determined by dividing the present value of the cash flow by the net present value of the observed instrument. Under the assumption of yearly compounding, Macaulay duration is given by

$$\frac{\sum T_i \times \frac{CF_i}{(1+r)^{T_i}}}{\sum \frac{CF_i}{(1+r)^{T_i}}}$$

where "CF_i" is the (absolute) amount of the i-th cash flow, "T_i" is the respective maturity of the i-th cash flow and "r" is the yield to maturity of this financial instrument.

- **Modified Duration** is a measure of price sensitivity, defined as the percentage derivative of price with respect to yield to maturity and compounding method. Under the same assumption, modified duration is given by

$$\frac{\sum T_i \times \frac{CF_i}{(1+r)^{T_i}}}{\sum \frac{CF_i}{(1+r)^{T_i}}} \times \frac{1}{(1+r)}$$

- **Fisher-Weil Duration**: If rates from a zero coupon yield curve are used instead of the yield to maturity, the Fisher-Weil duration is calculated with the same formula instead of Macaulay duration.

Flow Generation

The following rule lines are available:

- **Periodic Fixed Amount Flow**
  Calculates principal payments that, in combination with interest payments, would compose a fixed periodic total of equal value in all periods (but with constantly changing principal and interest values).
  Principal calculation is done in two steps:
  First, the amortizing factor for n-th principal payment (A_n) is found:

$$A_n = \frac{1 - (1 + \frac{r}{f})^{t_n - T}}{1 - (1 + \frac{r}{f})^{-T}}$$

  Where "r" is the nominal interest rate, "f" is the payment frequency (number of principal payments in a year), "t_n" is the term of the n-th principal payment, and "T" is the term of the last principal payment determined by the maturity.
  Then, the amortizing factors are used to calculate the periodic principal payments (P_n):

$$P_n = N(A_{n-1} - A_n)$$

- **Periodic Fixed Even Flow**
  Calculates equal principal payments (hence "even flow") by dividing the total outstanding amount on the maturity date with the total number of payment periods.
- **Periodic Fixed Interest Rate Flow**
  Calculates principal cash flows as the simple product of a selected rate and selected start value, except for the last payment, which is the difference between start value and sum of previous principal payments.
- **Periodic Fixed Rate Flow**
  This line type's output, principal cash flow, is determined by subtracting (1) interest payments from (2) total cash flows, except for the last payment which equals the remaining principal of the previous period:

1. Interest is the product of the input field *Rate* and the remaining principal of the previous period.
2. Total cash flow is the product of the input fields *Repayment Rate* and *Start Value*, with the exception of the last payment, which is the sum of the remaining principal and the last interest payment.

- **Periodic Fixed Value Flow**
  This line type's periodic cash flows are equal to the amount set in the input field *Value*. The exception to this is the last payment, which is the difference between the amount set in the input field *Start Value*, the sum of all previous periodic cash flows, and the amount set in the input field *End Value*.
- **Single Flow**
  Line types described above generate multiple output rows as principal and interest calculations across all periods. The *Single Flow* line type generates only one output row, which makes it suitable for special or atypical cash flows like balloon/bullet payments, administration fees, disbursements, and so on.

Flow Merge

The following line type is available:

- **Include Events to Flows**
  Joins an instrument's master data (for example, origination amount and date, currency) from one function with its tripartite event data (cash flow amount/type/date) from another function into one data record (hence "flow merge").

Market Interest Rate

> **i** Note
>
> For easier understanding, line type descriptions are given in the order of the calculation steps (they are not listed in alphabetic order).
>
> Some line types use continuous compounding to depict continuous and infinite reinvestment of interest, rather than monthly, quarterly, or annual compounding. Equations with continuous compounding use natural logarithms (for example, y = ln(x)) or inverse of natural log (x = exp(y)).

The following line types are available:

- **Effective Yield Rate**
  Calculates continuous effective interest rate with respect to a given set of future cash flows associated to a financial position in two steps. In the first step, it calculates the internal rRate of return (IRR) as the root of the following equation:

$$P_0 = \sum_{i=1}^{n} \frac{P_i}{(1+r)^{\frac{T_i}{d}}}$$

  where:
  - "P_i" is the i-th (re)payment including principal and interest
  - "T_i" is the term of the i-th payment counted in days, for example, the days from starting date to i-th payment counted regarding a given interest calculation method
  - "d" is the total number of days in a year regarding the given day count convention. For example, d = 360 for Act/360, 30/360; d = 365 for Act/365, and so on.

In the second and last step, the system calculates and displays the continuous effective interest rate "r^c_{eff}", based on the assumption of continuous compounding, as determined by the following formula:

$$\left(r^c_{eff}\right) = \ln(1 + r_{eff})$$

where "ln" is the natural logarithm.

- **Effective Capital over Time**
  Calculates the effective capital as the difference between the continually compounded initial capital (inflow) and the sum of subsequent repayments (outflows). Each period's effective capital is the difference between the previous period's effective capital (that is continually compounded) and the current period's repayment. The following equation resembles this relationship:

$$ECOT_i = ECOT_{i-1} \times e^{\left(r^c_{eff} \times \frac{T_i - T_{i-1}}{d}\right)} - P_i$$

  where i = 1,...,n and "$ECOT_0$" is the initial amount invested (for example, the total amount of a loan).
- **Capital Growth**
  Capital growth for a given period equals the ratio of (i) the product of the previous period's effective capital and the continuous effective rate for the elapsed period (hereinafter numerator) and (ii) the continuous effective interest rate (hereinafter denominator). The calculation is therefore similar to that of perpetual annuity: periodic income (numerator: product of effective capital and effective yield rate for the elapsed period) is divided by the effective yield rate (denominator).

$$CG_i = ECOT_{i-1} \times \frac{e^{\left(r^c_{eff} \times \frac{T_i - T_{i-1}}{d}\right)} - 1}{r^c_{eff}}$$

  where i = 1,...,n. By convention $CG_1 = 0$.
- **Capital Growth Net Present Value**
  Returns the simple product of an already calculated discount factor and capital growth to generate present values of capital growth for each period.
- **Net Present Value**
  Returns the simple product of an already calculated discount factor and cash flows to generate the present value of future cash flows.
- **Net Present Value Sum**
  Returns the sum of the present value of cash flows, which is the NPV of cash flows.
- **Margin Spread**
  Determines margin spread as the ratio of cash flow NPV and capital growth NPV. Margin spread is the rate that is usually charged above FTP rate to ensure that each investment (for example, an extended loan) generates positive returns. From a financial institutions's perspective, margin spread is the difference between the borrowing and lending rates in deposits or the difference between the cost of borrowing and return from lending.
- **FTP Rate**
  Determines the funds transfer pricing rate as the difference between continuous effective interest rate and margin spread. FTP is the rate at which a bank extends (or accepts) loans to (or from) its internal departments.
- **Market Interest Rate**
  This line type contains all of the Net Present Value approach's outputs as described above.

Matched Maturity

Matched Maturity, also known as "Term Weighted Matched Maturity", calculates the FTP rate by matching principal repayments' due dates to the rates on the marginal cost of funds curve (or. FTP curve) of same maturities, while treating the product of corresponding terms and principal repayments as weights:

$$\frac{\sum_{i=1}^{n} P_i \times T_i \times r_i}{\sum_{i=1}^{n} P_i \times T_i}$$

where "P_i" is the i-th principal payment amount relating to the i-th term "T_i" and "r_i" is the prevailing interest rate relating to "T_i" retrieved from the specified FTP rate curve, and "n" is the total number of principal payments.

> **i** Note
>
> Note on terminology: *Cost of Funds Curve* (usually an index like the LIBOR swap, FHLB funding, or local national government treasury curve) contains reference rates at which financial institutions presumably can raise debt. *Marginal Cost of Funds Curve* (or FTP Curve) modifies reference rates to account for incremental costs (banks bear) to get new funding because of their credit rating, liquidity (borrowing's size/term), and so on.

Use "Rate Modeling" or "Interpolation" rule types to generate the FTP curve, the main prerequisite of the Matched Maturity approach.

Rate Modeling

The following rule lines are available:

- **Daycount**
  Calculates the number of days between two dates: starting date(s) of input field *Date* and ending date(s) of input field *Curve Date*. The calculation also depends on daycount basis, like actual/actual, 30/360, and so on.
- Forward Rate
  Derives forward rates from spot rates of a selected yield curve by non-arbitrage principle using the equation below:

  $$\frac{(1 + r_{x+y})^{t_{x+y}}}{(1 + r_x)^{t_x}} - 1$$

  where
  - "r_{x+y}" is the spot rate of a zero-coupon bond of longer maturity term "t_{x+y}"
  - "r_x" is the spot rate of a zero-coupon bond of shorter maturity term "t_x".

  The following table compares derivation and application of spot and forward rates:

Spot Rates	Forward Rates
Zero-coupon treasury yields	Derived from spot rates
Today's price (or interest rate) of immediate transaction	Today's price (or interest rate) of transaction to occur in the future

Spot Rates	Forward Rates
Discounts a future cash flow to the present date	Discounts a distant future cash flow to a closer future date
Example: 3-month and 12-month zero-coupon rates	Example: 9-month rate expected 3 months from now (implied by 3-month and 12-month zero-coupon rates)

Non-arbitrage principle means that the forward rate shall equal the future spot rate. For example, a longer-term spot rate (for eample 12 months) shall equal compounded return of the shorter-term spot rate (for example 3 months) and the remaining-term forward rate (for example 9 months).

- **Lookup Rate by Interpolation**
  Linearly interpolates rates of a yield curve (of the *Lookup* tab) on dates from the input field *Date* (of the *Input* function) and assigns them to the Input function's instrument(s) of matching characteristic values for curve ID, validity date, and currency.
- **Periodic Fixed Interest**
  Produces periodic interest ordinarily using a set of cash outflows (for example disbursements) and cash inflows (principal repayments) as input. Specifically, the calculation applies the **fixed** periodic interest rate to the instrument's remaining value (outstanding balance) – usually a difference between initial disbursement and the sum of subsequent principal repayments.
- Periodic Variable Interest
  Produces periodic interest, ordinarily using a set of cash outflows (for example disbursements) and cash inflows (principal repayments) as input. Specifically, the calculation applies the **variable** periodic interest rates looked up from a certain interest rate curve to the instrument's remaining value (outstanding balance) – usually a difference between initial disbursement and the sum of subsequent principal repayments.

Strip Funding

Strip Funding calculates the FTP rate as the ratio of (i) NPV of principal payments discounted using maturity matching funding rates by annual compound method and (ii) the sum of discounted and term-accrued principal balances remaining at each period.

$$\frac{B_0 - \sum_{i=1}^{N} P_i \times d_i}{\sum_{i=1}^{N} B_{i-1} \times \Lambda_{i-1,i} \times d_i}$$

where "B_0" is the initial balance on FTP pricing date, "B_i" is the outstanding closing balance of the payment period i. "P_i" is the principal payment amount of this payment period. "$Lambda_{i-1}$" is the accrual factor in year for the whole period i. And "d_i" is the discount factor derived from a given FTP reference curve by annual compound method.

> **i** Note
>
> Note on terminology: "Strip Funding" is a widely accepted method of separately valuating each principal cash flow (in case of maturing products) or each component (in case of non-maturing products) to

determine the FTP rate. Each principal cash flow or component is, “stripped” from the rest and assigned a funding rate, and so on. The rule types *Matching Maturity*, *Strip Funding* and *Weighted Average Rate* apply this general concept using different equations to generate FTP rates.

Weighted Average Rate

This approach entails four steps:

1. Splits a financial position into components according to a certain logic (like differing behavioral tendencies of position’s parts). The chosen logic mandates the remaining steps.
2. Assigns weighting factors (percentages or amounts) to each part.
3. Selects a suitable marginal cost of funds curve and assigns relevant funding rates to each component.
4. Calculates FTP (WAR) as the ratio of
   - product sum of factors (f_i) and corresponding funding rates (r_i) and
   - sum of factors (f_i) as shown in the equation below.

> **i** Note
>
> Only step four is solely set up and executed in this line type; the first three steps are executed in other rule types in SAP Profitability and Performance Management (for example, *Join* or *Interpolation*.).

$$\frac{\sum f_i \times r_i}{\sum f_i}$$

This rule line type calculates the average funding rate from an FTP curve weighted by term – a special variant of the term-weighted matched maturity method. It is also a non-cash-flow transfer pricing method for non-maturity positions like checking, savings, money market, and credit card accounts. These kinds of non-contractual cash flow balances can behave both as long and short maturities, and can be split accordingly and assigned a respective long-term and short-term interest rate.

Procedure

Function Access

Follow the steps below to access the *Funds Transfer Pricing* function:

1. In the client where SAP Profitability and Performance Management is installed, choose *SAP Menu* > *Profitability and Performance Management* > *Modeling* > *Start My Environments*.
2. The *Environment* screen appears in a separate browser window. Choose an existing environment and continue. Within the environment, you can set up the newly added function.

Function Configuration

Follow the steps below to configure the *Funds Transfer Pricing* function:

1. In edit mode, configure the following required fields in the header.
   - *Result Handling*
   - *Suppress initial Result*

- *Include original Input Data*
- *Result Model Table*

2. Define the input function to be used on the *Input* tab.

> **i Note**
>
> For more information about the *Input* tab, see Input [page 11].

3. Define the function to be used on the *Lookup* tab, if needed.

> **i Note**
>
> For more information about the *Lookup* tab, see Lookup [page 12].

4. Define the fields to be used on the *Signature* tab.

> **i Note**
>
> For more information about the *Signature* tab, see Signature [page 13].

5. On the *Rules* tab, you can define the rule and line types of the *Funds Transfer Pricing* function. For more information about the available rule types, see the *Rule Types* section above.
   Proceed as follows to configure the rules:
   1. On the *Rule* tab of your *Funds Transfer Pricing* function, choose + (*Add*).
   2. The *Add Details* screen is displayed. Enter the following information:
      - *Add Level*
      - *Rule*
      - *Rule Type*
      - *Description*
   3. Choose *OK*.
   4. Select the created rule.
   5. On the *Rule Lines* tab of the created rule, choose + (*Add*).
   6. The *Add Details* screen is displayed. Enter the required information.
   7. Select the created rule line and enter the required information.

      > **i Note**
      >
      > Depending on the chosen rule type the rule line shows different fields.

6. Define the checks to be used on the *Checks* tab, if needed.

> **i Note**
>
> For more information about the *Checks* tab, see Checks [page 17].

7. Choose *Save* and then choose *Activate*.

Related Information

- For more information about common aspects of SAP Profitability and Performance Management functions, see Concepts for Key Users [page 6].

- For more information on how to add and remove functions in SAP Profitability and Performance Management environments, see Function Hierarchy [page 63].

1.4.2.7.1 Example: Series Generation

Input Data

Position ID	Increment	Minimum	Maximum	Element Number
001	1	1	500	0

The function generates a series of values starting from “1” with an increment equal to “1” until the element number reaches the maximum, which is set to “500”.

Result

Position ID	Increment	Minimum	Maximum	Element Number
001	1	1	500	1
001	1	1	500	2
001	1	1	500	3
001	1	1	500	4
001	1	1	500	5
001	1	1	500	6
001	1	1	500	7
001	1	1	500	8
001	1	1	500	9
001	1	1	500	...
001	1	1	500	500

1.4.2.7.2 Example: Formula

Input Data

Position ID	Term (m)	Cash Flow Principal	Currency	Term Unit	Interest Calculation Method	Interest Rate (%)	Discount Factor
L040000000	0.000	10,000.000	USD	Day	360	0.000	1.000
L040000000	31.000	302.000	USD	Day	360	1.893	0.998
L040000000	59.000	302.000	USD	Day	360	2.033	0.997
L040000000	90.000	302.000	USD	Day	360	2.134	0.995
L040000000	120.000	302.000	USD	Day	360	2.226	0.993
L040000000	151.000	302.000	USD	Day	360	2.319	0.990
L040000000	181.000	302.000	USD	Day	360	2.408	0.988
L040000000	212.000	302.000	USD	Day	360	2.499	0.986
L040000000	243.000	302.000	USD	Day	360	2.590	0.983
L040000000	273.000	302.000	USD	Day	360	2.678	0.980
L040000000	304.000	302.000	USD	Day	360	2.769	0.977
L040000000	334.000	302.000	USD	Day	360	2.856	0.974
L040000000	365.000	302.000	USD	Day	360	2.947	0.971
L040000000	396.000	302.000	USD	Day	360	2.971	0.968
L040000000	425.000	302.000	USD	Day	360	2.991	0.966
L040000000	456.000	302.000	USD	Day	360	3.012	0.963
L040000000	486.000	302.000	USD	Day	360	3.033	0.960

Position ID	Term (m)	Cash Flow Principal	Currency	Term Unit	Interest Calculation Method	Interest Rate (%)	Discount Factor
L040000000	517.000	302.000	USD	Day	360	3.054	0.958
L040000000	547.000	302.000	USD	Day	360	3.074	0.955
L040000000	578.000	302.000	USD	Day	360	3.096	0.952

Configuration

Rule	Description	Rule Type	State	Rule Grouping Fields	Rule Ordering Fields	Selection
R001	Key Figure Formula	Key Figure Formula	Active			

Line	Description	Formula	Result
L001	Key Figure Formula	Cash Flow Principal * 0.5	(Net) Present Value

Based on the configuration, the key figure that the formula applies is “Cash Flow Principal”. The system applies the formula “Cash Flow Principal * 0.5” for each data entry and stores it in the additional column *(Net) Present Value*.

Expected Result

Position ID	Term (m)	Cash Flow Principal	Currency	Term Unit	Interest Calculation Method	Interest Rate (%)	Discount Factor	(Net) Present Value
L040000000	0.000	10,000.000	USD	Day	360	0.000	1.000	5,000.00
L040000000	31.000	302.000	USD	Day	360	1.893	0.998	151.00
L040000000	59.000	302.000	USD	Day	360	2.033	0.997	151.00
L040000000	90.000	302.000	USD	Day	360	2.134	0.995	151.00
L040000000	120.000	302.000	USD	Day	360	2.226	0.993	151.00

Position ID	Term (m)	Cash Flow Principal	Currency	Term Unit	Interest Calculation Method	Interest Rate (%)	Discount Factor	(Net) Present Value
L0400000 00	151.000	302.000	USD	Day	360	2.319	0.990	151.00
L0400000 00	181.000	302.000	USD	Day	360	2.408	0.988	151.00
L0400000 00	212.000	302.000	USD	Day	360	2.499	0.986	151.00
L0400000 00	243.000	302.000	USD	Day	360	2.590	0.983	151.00
L0400000 00	273.000	302.000	USD	Day	360	2.678	0.980	151.00
L0400000 00	304.000	302.000	USD	Day	360	2.769	0.977	151.00
L0400000 00	334.000	302.000	USD	Day	360	2.856	0.974	151.00
L0400000 00	365.000	302.000	USD	Day	360	2.947	0.971	151.00
L0400000 00	396.000	302.000	USD	Day	360	2.971	0.968	151.00
L0400000 00	425.000	302.000	USD	Day	360	2.991	0.966	151.00
L0400000 00	456.000	302.000	USD	Day	360	3.012	0.963	151.00
L0400000 00	486.000	302.000	USD	Day	360	3.033	0.960	151.00
L0400000 00	517.000	302.000	USD	Day	360	3.054	0.958	151.00
L0400000 00	547.000	302.000	USD	Day	360	3.074	0.955	151.00
L0400000 00	578.000	302.000	USD	Day	360	3.096	0.952	151.00

1.4.2.7.3 Example: Running Total

Input Data

Yield Curve Type	Period	Interpolated Yield	Amount
Yield Curve	0	–0.0001537	100.0000
Yield Curve	1	0.0043374	100.4337
Yield Curve	2	0.0088285	101.7735
Yield Curve	3	0.0096286	102.9165
Yield Curve	4	0.0104287	104.2372
Yield Curve	5	0.011203	105.7284
Yield Curve	6	0.0119945	107.4160
Yield Curve	7	0.012786	109.3009
Yield Curve	8	0.0135775	111.3924
Yield Curve	9	0.0143518	113.6835
Yield Curve	10	0.0151261	116.1983
Yield Curve	11	0.0159004	118.9495
Yield Curve	12	0.0166747	121.9507
Yield Curve	13	0.017449	125.2173

Configuration

Rule	Description	Rule Type	State	Rule Grouping Fields	Rule Ordering Fields	Selection
R001	Running Total	Running Total	Active			

Line	Description	Value	Granularity	Term	Running Total
L001	Running Total	Amount	Yield Curve Type	Period	Running Total

Based on the configuration, the key figure to be aggregated for the running total is *Amount*. The system computes the total of all the data and stores it in the additional column *Running Total*.

Expected Result

Yield Curve Type	Period	Interpolated Yield	Amount	Running Total
Yield Curve	0	–0.0001537	100.0000	100.0000
Yield Curve	1	0.0043374	100.4337	200.4337
Yield Curve	2	0.0088285	101.7735	302.2072
Yield Curve	3	0.0096286	102.9165	405.1237
Yield Curve	4	0.0104287	104.2372	509.3609
Yield Curve	5	0.011203	105.7284	615.0893
Yield Curve	6	0.0119945	107.4160	722.5053
Yield Curve	7	0.012786	109.3009	831.8062
Yield Curve	8	0.0135775	111.3924	943.1987
Yield Curve	9	0.0143518	113.6835	1056.8822
Yield Curve	10	0.0151261	116.1983	1173.0805
Yield Curve	11	0.0159004	118.9495	1292.0300
Yield Curve	12	0.0166747	121.9507	1413.9806
Yield Curve	13	0.017449	125.2173	1539.1980

1.4.2.8 Join

Join is a data access function that brings together the results of two or more other functions based on defined rules.

Key Features

Header

In the header, you define the principal behavior of the join.

You can choose between different join types:

- Implicit Fields
  Fields coming from inputs are automatically considered during the join procedure. If the field is visible on multiple join rules (or inputs), you can use the *Auto Filling* function.
- Explicit Fields
  Fields of inputs are manually defined and maintained by the configurer. These fields will then be considered during processing.

> **Example**
>
> In the “Union All” join type, the modeler has to add all fields required for the output even if they are not part of the input data. The fields must be added on each subview of the rules to be unioned before activation.

The modeler can control the join results with the *Auto Filling* option. The following settings are possible:

- “No”: If a field is defined in a join rule in multiple inputs, the field content is taken from the first input that contains the field.
- “If Null then First to Last”: The first non-null value is taken and if all values are null, the initial value is returned for that field.
- “If Null/Initial then First to Last”: The first non-null and non-initial value is taken and if all values are null or initial, an initial value is returned for that field.

> **Note**
>
> **Null vs Initial**
>
> In SAP Profitability and Performance Management UI, both NULL and INITIAL are represented by an EMPTY cell for the characteristic and “0” for the key figure. The modeler must take care when processing input data with these values, because the response is different:
>
> - NULL (?) does not have any value and does not occupy memory. This can be achieved if a field was not assigned with values initially.
> - INITIAL (' ') has a value and so occupies memory. In SAP Profitability and Performance Management a cell can have an INITIAL value if it was assigned with an empty record.

Input Tab

When the *Input* tab is assigned with an input function, a new dialog box that contains the complex selection and normal selection is available. In the complex selection, you can add a more sophisticated statement in order to retrieve the data.

Rules

Each join rule semantically defines the reading of a specific input.

Hierarchical join rules are also supported by assigning higher levels. The hierarchy of levels is resolved starting with the highest level, feeding as input to the lower levels and ending with level 0.

The following rule types are available:

1. *From*: This is always the first rule of a level.
2. *Left Outer Join*: This join type returns all rows from the rule above, and the columns and rows from this rule, where the predicates match.
3. *Inner Join*: This join type returns all rows when there is at least one predicate match in the rule above and this rule.
4. *Full Outer Join*: This join type returns all (matched or unmatched) rows from both the rule above and this rule.
5. *Cross Join*: This join type returns the cartesian product of the rule above and this rule.
6. *Union All*: This join behaves in the same way as a union, but duplicate records are not removed.
7. *Lookup*: Looks up fields and fills them in the first non-lookup rule above where the predicates match. At least one field needs to be defined as a lookup field.

8. *Lookup Auto Predicate*: Looks up fields and fills them in the first non-lookup rule where all common fields match those of the input function which is set on the *Rule* tab (this means, Field 1 of Rule 1 = Field 1 of Rule 2). At least one field needs to be defined as a lookup field.

Sub View

You can define further selections, formulas, aggregations and sorting orders for each rule.

Complex Selections

If required, you can define complex selections using formulas and SQL functions.

Join Predicates

You can define the predicate conditions for the matching for join and lookup rules here.

Complex Predicates

If required, you can enter complex on-predicates for join and lookup rules here using formulas and SQL functions.

> **i** Note
>
> Fields coming from inputs are automatically considered during the join procedure. If the field is visible onThese settings are only relevant for multiple join rules in which either non-null or non-initial values of the same field are considered and returned for that field. If there is only one rule, the Auto Filling options are not relevant.

Usage of HANA HINT in complex selections could help in scenarios where a join function execution runs into an out of memory dump or is very slow. Refer to the HANA help for HINT to evaluate which HINT could help in your implementation-specific scenario (rules at the same level or multi-level rules which lead to subqueries, formula/selection being used).

Procedure

Function Access

Follow the steps below to access the *Join* function:

1. In the client where SAP Profitability and Performance Management is installed, choose *SAP Menu* > *Profitability and Performance Management* > *Modeling* > *Start My Environments* .
2. The *Environment* screen appears in a separate browser window. Choose an existing environment and continue. Within the environment, you can set up the newly added function.

Function Configuration

Follow the steps below to configure the *Join* function:

1. In edit mode, configure the following required fields in the header. For more information about the header, see the *Key Features* section above.
   - *Join Type*
   - *Result Handling*

- *Suppress initial Result*
- *Include original Input Data*
- *Result Model Table*

2. Define the input function to be used on the *Input* tab.

> **i Note**
>
> For more information about the *Input* tab, see Input [page 11].

3. Define the fields to be used on the *Signature* tab.

> **i Note**
>
> For more information about the *Signature* tab, see Signature [page 13].

4. On the *Rules* tab, you can define the rule types of the *Join* function.
   Proceed as follows to configure the rules:
   1. On the *Rule* tab of your *Join* function, choose + (*Add*).
   2. The *Add Details* screen is displayed. Enter the following information:
      - *Add Level*
      - *Rule*
      - *Rule Type*
      - *Input Description*
   3. Choose *OK*.
   4. Select the created rule.
   5. Define the input function to be used.
   6. On the *Subview* tab created rule, choose + (*Add*).
   7. The *Fields* screen appears where you can select the fields to be used.
   8. Choose *OK*.
   9. Follow the steps below if you want to setup or apply a formula for a specific field (optional):
      1. Select the field and choose *f* (*Formula*).
      2. On the *Formula* screen which is displayed next, enter the formula.
      3. Choose *OK*.
   10. Follow the steps below if you want to setup or apply a selection condition for a specific field (optional):
      1. Select the field and choose *Selection Condition*.
      2. On the *Select Condition* screen which is displayed next, enter the condition.
      3. Choose *OK*.
   11. On the *Complex Selections* tab of the created rule, you can directly define complex selections using formulas and SQL functions.
   12. On the *Join Predicates* tab of the created rule, choose + (*Add*).
   13. The system automatically adds a new line entry. Enter the following information here:
      - *Field*
      - *Comparison*
      - *Join Rule*
      - *Join Field*
   14. On the *Complex Predicates* tab of the created rule, you can directly enter “on” predicates for join and lookup rules using formulas and SQL functions.

5. Define the checks to be used on the *Checks* tab, if needed.

> **i Note**
>
> For more information about the *Checks* tab, see Checks [page 17].

6. Choose *Save* and then choose *Activate*.

Related Information

- For more information about the autofilling option of the *Join* function, see Handling Null and Empty Values in Join Function using Autofilling Option.
- For more information about common aspects of SAP Profitability and Performance Management functions, see Concepts for Key Users [page 6].
- For more information on how to add and remove functions in SAP Profitability and Performance Management environments, see Function Hierarchy [page 63].

1.4.2.8.1 Full Outer Join Special Scenarios for Auto Filling Field

1.4.2.8.1.1 Example: Full Outer Join – Auto Filling Set to "No"

This scenario shows how the rule type “Full Outer Join” merges two input tables with one join predicate as a bridge.

Input Tables

Product - Material Table

Product Code	Material Code	Request Order
P0001	M1011	5
P0001	M1010	2
P0002	M1009	1
P0002	M1011	3
P0005	M1012	10
P0005	M1011	30

JO - Product Table

Product Code	Product	Price	Unit
P0001	Shoe	75	EUR
P0002	Watch	300	EUR
P0003	Shirt	80	EUR
P0004	Shorts	20	EUR

This scenario executes a Full Outer Join that is based on the join predicate.

In this case, *Product Code* is the predicate used for both rules and all items with product code “P0001” and “P0002” are added in the final results.

Interim Result (Product - Material Table and JO - Product Table)

Product - Material Table

Product Code	Material Code	Request Order
P0001	M1011	5
P0001	M1010	2
P0002	M1009	1
P0002	M1011	3

JO - Product Table

Product Code	Product	Price	Unit
P0001	Shoe	75	EUR
P0002	Watch	300	EUR

The following table shows parts of the Full Outer Join table. However, since they contain "?" or null values, they are not included in the final results.

Product Code	Material Code	Request Order
P0005	M1012	10
P0005	M1011	30
	?	?
	?	?

Product Code	Product	Price	Unit
	?	?	?
	?	?	?
P0003	Shirt	80	EUR
P0004	Shorts	20	EUR

The system does not return the rest of the non-null and/or non-initial values unless the auto filling setting is set to "If Null/Initial then First to Last".

The null values are caught by an error handler that informs you if there are null values for the join of the first rule with the fields *Material Code*, *Request Order*, *Price* and *Unit* for the product codes that are not included in the output.

Expected Result

Material Codw	Request Order	Product Code	Product	Price	Unit
M1011	5	P0001	Shoe	75	EUR
M1010	2	P0001	Shoe	75	EUR
M1009	1	P0002	Watch	300	EUR
M1011	3	P0002	Watch	300	EUR

Returns values for matching rows ("P0001" and "P0002") based on the join predicates set for the field *Product Code* (`PROD_CODE`).

1.4.2.8.1.2 Example: If Null/Initial Then First to Last

This scenario shows the enrichment capability of the *Join* function to highlight the effect of Auto Filling set to "Null/Initial then First to Last".

That means, the system substitutes null and initial values (for *Characteristic* it is whitespace (' ') and for *Key Figure* it is zero ('0')) by the next non-null value of the succeeding table having the same field. If this field only has initial/null values, the system just sets an initial value.

Input Tables

Legend: '' is considered as an initial or empty input

JO – Product / Customer in US

Product	Customer	Amount
PROD01	US_CUST01	200
PROD04	US-CUST04	100
PROD06	US_CUST06	300
PROD07	''	100
PROD08	''	200
PROD09	US_CUST09	300

JO – Product / Customer in DE

Product	Customer	Price
PROD01	DE_CUST01	120
PROD04	DE_CUST04	60
PROD05	DE_CUST05	180
PROD07	DE_CUST07	60
PROD08	DE_CUST08	120
PROD10	DE_CUST10	180

The system takes the first non-null and non-initial value and if all values are null or initial, it returns an initialized value, empty or blank for a Character (CHAR) field and "0" for a *Key Figure* field.

Interim Results

Product	Customer	Amount	Price
PROD01	US_CUST01	200	120
PROD04	US_CUST04	100	60
PROD05	DE_CUST05	?	180
PROD06	US_CUST06	300	?
PROD07	''	100	60
PROD08	''	200	120
PROD09	US_CUST09	300	?
PROD10	DE_CUST10	?	180

In PROD05 row, we had our first null value (?). Since we only have two tables we won't be able to look further for another initial value. Since all values are null for the *Amount* field, the system returns an initialized value. For *Key Figure*, it will be "0". The same scenario will be encountered for PROD06, PROD09 and PROD10.

In PROD07 row, we had our first initial value (''), as you can see we look at the next table for the same *Customer* field. Since the next value for Customer is "DE_CUST07", it will be the value for *Customer* field at PROD07. The same is true for PROD08 scenario, which will have a value of "DE_CUST08".

Expected Result

Product	Customer	Amount	Price
PROD01	US_CUST01	200	120
PROD04	US_CUST04	100	60
PROD05	DE_CUST05	0	180
PROD06	US_CUST06	300	0
PROD07	DE_CUST07	100	60
PROD08	DE_CUST08	200	120
PROD09	US_CUST09	300	0
PROD10	DE_CUST10	0	180

1.4.2.8.1.3 Example: If Null Then First to Last

This scenario shows the enrichment capability of the *Join* function to highlight the effect of Auto Filling set to "Null then First to Last".

That means, the system substitutes null values (' ') of the fields by the next non-null value of the succeeding table having the same field. If this field only has null values, the initial value is set.

Input Tables
Legend: " is considered as an initial or empty input

JO – Product / Customer in US

Product	Customer	Amount
PROD01	US_CUST01	200
PROD04	US-CUST04	100
PROD06	US_CUST06	300
PROD07	"	100
PROD08	"	200
PROD09	US_CUST09	300

JO – Product / Customer in DE

Product	Customer	Price
PROD01	DE_CUST01	120
PROD04	DE_CUST04	60
PROD05	DE_CUST05	180
PROD07	DE_CUST07	60
PROD08	DE_CUST08	120
PROD10	DE_CUST10	180

The system takes the first non-null value and if all values are null, it returns the initial value for that field.

Interim Results

Product	Customer	Amount	Price
PROD01	US_CUST01	200	120
PROD04	US_CUST04	100	60
PROD05	DE_CUST05	?	180
PROD06	US_CUST06	300	?

Product	Customer	Amount	Price
PROD07	''	100	60
PROD08	''	200	120
PROD09	US_CUST09	300	?
PROD10	DE_CUST10	?	180

In the row PROD05, we have our first null value (?). Since all values are null for the *Amount* field, the system returns an initialized value. For *Key Figure*, the value is 0. The same scenario applies to PROD06, PROD09 and PROD10.

In the row PROD07, we have our first initial value (''). Since this initial value ('') is a non-null value, it is used as the result. The same is true for the scenario PROD08, which has a value of empty or blank because the type is "Character".

Expected Result

Product	Customer	Amount	Price
PROD01	US_CUST01	200	120
PROD04	US_CUST04	100	60
PROD05	DE_CUST05	0	180
PROD06	US_CUST06	300	0
PROD07		100	60
PROD08		200	120
PROD09	US_CUST09	300	0
PROD10	DE_CUST10	0	180

1.4.2.8.2 Example: Inner Join

This scenario shows how the inner join merges the two input tables and returns values that match with the set join predicate.

Input Tables

Product - Material Table

Product Code	Material Code	Request Order
P0001	M1011	5
P0001	M1010	2
P0002	M1009	1
P0002	M1011	3
P0005	M1012	10
P0005	M1011	30

Product Table

Product Code	Product	Price	Currency
P0001	Shoe	75	EUR
P0002	Watch	300	EUR
P0003	Shirt	80	EUR
P0004	Shorts	20	EUR

Returns all values with corresponding matches based on the join predicates for product code “P0001” and “P0002”.

Product, *Price* and *Currency* which correspond to each *Product Code* will be distributed to each product code resulting in the below table.

Interim Results (Product - Material Table and Product Table)

Product Code	Material Code	Request Order	Product	Price	Currency
P0001	M1011	5	Shoe	75	EUR
P0001	M1010	2	Shoe	75	EUR
P0002	M1009	1	Watch	300	EUR
P0002	M1011	3	Watch	300	EUR

1.4.2.8.3 Example: Left Outer Join

This scenario shows how the left outer join merges the two input tables and returns all values from the left table and all corresponding matches based on the join predicates. The *Auto Filling* option of two join tables is set to "If Null/Initial then First to Last".

Input Tables

Product - Material Table

Product Code	Material Code	Request Order
P0001	M1011	5
P0001	M1010	2
P0002	M1009	1
P0002	M1011	3
P0005	M1012	10
P0005	M1011	30

Product Table

Product Code	Product	Price	Currency
P0001	Shoe	75	EUR
P0002	Watch	300	EUR
P0003	Shirt	80	EUR
P0004	Shorts	20	EUR

Returns all values from the left table and all corresponding matches based on the join predicates for product codes P0001 and P0002.

Expected Result

Product Code	Material Code	Request Order	Product	Price	Currency
P0001	M1011	5	Shoe	75	EUR
P0001	M1010	2	Shoe	75	EUR
P0002	M1009	1	Watch	300	EUR
P0002	M1011	3	Watch	300	EUR
P0005	M1012	10		0	
P0005	M1012	30		0	

1.4.2.8.4 Example: Cross Join Implicit

This scenario merges the three input tables and returns all row combinations from the records of the inputs.

Input Tables

Material Table

Material Code	Material	Cost per Order	Unit
M1011	Leather	10	USD
M1012	Thread	5	USD

Product Table

Product Code	Product
P0001	Shoe
P0002	Watch

Order Table

Branch	Order
BR001	10
BR002	20

Interim Result of Material Table and Product Table (MatPro Table)

Material Code	Material	Cost per Order	Unit	Product Code	Product
M1011	Leather	10	USD	P0001	Shoe
M1012	Thread	5	USD	P0001	Shoe
M1011	Leather	10	USD	P0002	Watch
M1012	Thread	5	USD	P0002	Watch

Returns material code M1011 and M1012 for product code P0001.

Returns material code M1011 and M1012 for product code P0002.

The Cross Join produces an interim result set which is the number of rows in the first table multiplied by the number of rows in the second table.

Expected Result

Material Code	Material	Cost per Order	Unit	Product Code	Product	Branch	Order
M1011	Leather	10	USD	P0001	Shoe	BR001	10
M1011	Leather	10	USD	P0001	Shoe	BR002	20
M1011	Leather	10	USD	P0002	Watch	BR001	10
M1011	Leather	10	USD	P0002	Watch	BR002	20
M1012	Thread	5	USD	P0001	Shoe	BR001	10
M1012	Thread	5	USD	P0001	Shoe	BR002	20
M1012	Thread	5	USD	P0001	Watch	BR001	10
M1012	Thread	5	USD	P0002	Watch	BR002	20

The Cross Join produces a result set which is the number of rows in the interim table result multiplied by the number of rows on the third table.

In this result, we are identifying the number of orders, per product and per branch by cross referencing from *Material Table* to *Product Table* and *Order Table*.

1.4.2.8.5 Example: Union All

This scenario shows how the *Join* function merges the two input tables based on a set of rules.

Input Tables

JO - Material Table

Material Code	Material	Cost per Order	Currency
M1001	Paper	3	USD
M1002	Plastic	3	USD
M1003	Wax	4	USD
M1006	Botton	1	USD
M1007	Cotton	10	USD
M1009	Glass	50	USD
M1010	Lace	5	USD
M1011	Leather	10	USD
M1012	Thread	5	USD

JO - Product Table

Product Code	Product	Price	Currency
P0001	Shoe	75	EUR
P0002	Watch	300	EUR
P0003	Shirt	80	EUR
P0004	Shorts	20	EUR

Filter the tables first so that you have only the required entries:

- For the Material Table, we set the condition to M1001, M1002 and M1003.
- For the Product Table, we set the condition to P0001, P0002 and P0003.

Based on the selection conditions we set in the subview of each rule, we will have the following tables:

Interim Result

Material Table

Material Code	Material	Cost per Order	Currency
M1001	Paper	3.00	USD
M1002	Plastic	3.00	USD
M1003	Wax	4.00	USD

Product Table

Product Code	Product	Price	Currency
P0001	Shoe	75	EUR
P0002	Watch	300	EUR
P0003	Shirt	80	EUR

If we are using an explicit type of join, the fields that we define in the subview of each rule will be the output. However, we need to specify all the fields that we need in the subview of each rule because we need to have the same fields across the subview for the explicit view to work.

Union All is used to combine the result sets of two or more tables. It does not remove duplicate rows and all rows are returned.

Expected Result

Material Code	Material	Product Code	Product	Cost per Order	Price	Currency
M1001	Paper			3	0	USD
M1002	Plastic			3	0	USD
M1003	Wax			4	0	USD
		P0001	Shoe	0	75	EUR
		P0002	Watch	0	300	EUR
		P0003	Shirt	0	80	EUR

1.4.2.8.6 Example: Lookup Auto Predicates

For this scenario, corresponding material is retrieved and displayed for every matching entry in the field *Material Code*.

Level 1 Processing

Product - Material Table will be enriched as the corresponding *Material* will be retrieved and displayed for every matching entry in the field *Material Code* (`MAT_CODE`).

> **i** Note
>
> The system resolves the hierarchy of levels starting with the highest level, feeding as an input to the lower levels and ending with level 0.

Product - Material Table (Level 1, From)

Product Code	Material Code	# Request Order
P0001	M1011	5
P0001	M1010	2
P0002	M1009	1
P0002	M1011	3
P0005	M1012	10
P0006	M1011	30

Material Table (Level 1, Lookup Auto Predicate)

Material Code	Material	Cost per Order	Unit
M1001	Paper	3	USD
M1002	Plastic	3	USD
M1003	Wax	4	USD
M1006	Botton	1	USD
M1007	Cotton	10	USD
M1009	Glass	50	USD
M1010	Lace	5	USD
M1011	Leather	10	USD
M1012	Thread	5	USD

Interim Result (Level 1)

Product Code	Materials Code	# Request Order	Material
P0001	M1011	5	Leather
P0001	M1010	2	Lace
P0002	M1009	1	Glass
P0002	M1011	3	Leather
P0005	M1012	10	Thread
P0006	M1011	30	Leather

Level 0 Processing

Level 0 Processing

The product table declared in the first rule (“From”) will now perform a Left Outer Join (for every matched product code (`PROD_CODE`) entry) with the Level 1 result (enhanced Product - Material Table) since result processed from a higher level will be considered as an input for the lower level.

> **i Note**
>
> Setting the Product - Material Table as an input function for the second rule will not affect the results of the join since the system automatically detects that the input will be coming from the enhanced Product - Material Table.

Product Table (Level 0, From)

Product Code	Product	Price	Unit
P0001	Shoe	75	EUR
P0002	Watch	300	EUR
P0003	Shirt	80	EUR
P0004	Shorts	20	EUR

Interim Result (Level 1)

Product Code	Materials Code	# Request Order	Material
P0001	M1011	5	Leather
P0001	M1010	2	Lace
P0002	M1009	1	Glass
P0002	M1011	3	Leather
P0005	M1012	10	Thread
P0006	M1011	30	Leather

Expected Result

Product Code	Product	Price	Unit	Material Code	Request Order	Material
P0001	Shoe	75	EUR	M1011	5	Leather
P0001	Shoe	75	EUR	M1010	2	Lace
P0002	Watch	300	EUR	M1009	1	Glass
P0002	Watch	300	EUR	M1011	3	Leather

1.4.2.9 Transfer Structure

Transfer Structure is a data enrichment function that can be used to transpose data according to predefined condition fields and settings. If those conditions are not met, the Transfer Structure function retains the source data. The function provides a pivot and an unpivot option.

Key Features

Header

In the header, you define the principal behavior of the *Transfer Structure* function.

Transfer Structure Type:

- Transfer Structure [page 283]
- Reverse Transfer Structure [page 286]

The *Retain Fields* option is relevant for the Transfer Structure type:

- All Fields: All fields are retained.
- All Fields except *Selection & Action* > *Source Fields*: Selection condition fields and action > source fields are excluded from the result.

The *Aggregate Result* option is relevant for the Transfer Structure type:

- Group Characteristics: Key figures are automatically aggregated using all input characteristics as grouping fields.
- Group Characteristics and Key Figures: Key figures are automatically aggregated using all input characteristics and key figures as grouping fields.
- No Grouping: No aggregation takes place.

Rules

Each transfer structure rule semantically defines an if-then statement. The if part selects which subset of the input data the rule is relevant to. The then part is an action and contains a list of fields and values that need to be assigned.

Related Information

For more information about common aspects of SAP Profitability and Performance Management functions, see Concepts for Key Users [page 6].

1.4.2.9.1 Transfer Structure Type: Transfer Structure

This method transfers values to columns, sometimes also called “pivoting of data”. It is typically used to turn account-based data into costing-based data.

Procedure

Function Access

Follow the steps below to access the *Transfer Structure* function of type “Transfer Structure”:

1. In the client where SAP Profitability and Performance Management is installed, choose *SAP Menu* > *Profitability and Performance Management* > *Modeling* > *Start My Environments*.
2. The *Environment* screen appears in a separate browser window. Choose an existing environment and continue. Within the environment, you can set up the newly added function.

Function Configuration

Follow the steps below to configure the *Transfer Structure* function of type “Transfer Structure”:

1. In edit mode, configure the following required fields in the header. For more information about the header, see the *Key Features* section in Transfer Structure [page 282].
   - *Transfer Structure Type* = “Transfer Structure”
   - *Result Handling*
   - *Suppress initial Result*
   - *Ensure Distinct Result*
   - *Include original Input Data*
   - *Result Model Table*
   - *Retail Fields*
   - *Aggregate Result*
2. Define the input function to be used on the *Input* tab.

   > **i Note**
   >
   > For more information about the *Input* tab, see Input [page 11].

3. Define the fields to be used on the *Signature* tab.

   > **i Note**
   >
   > For more information about the *Signature* tab, see Signature [page 13].

4. On the *Rules* tab, each transfer structure rule defines semantically an if-then statement. The “if”-part selects which subset of the input data the rule is relevant to. The “then”-part is an action, and contains a list of fields and values that have to be assigned.
   Proceed as follows to configure the rules:
   1. On the *Rule* tab of your *Transfer Structure* function, choose + (*Add*).
   2. The *Add Details* screen is displayed. Enter the following information:

- *Rule*
- *Description*

3. Choose *OK*.
4. Select the created rule.
5. In the *Selection* section of the created rule, choose + (*Add*).
6. On the *Fields* screen that is displayed next, choose the field to be used and then choose *OK*.
7. Select the field and choose *Selection Condition*.
8. On the *Selection Condition* screen, define the selection condition to be used and choose *OK*.
9. In the *Action* section of the created rule, choose + (*Add*).
10. On the *Fields* screen that is displayed next, choose the field to be used and choose *OK*.
11. Select the field and choose *Formula*.
12. On the *Formula* screen that is displayed next, define the formula to be used and choose *OK*.

5. Define the checks to be used on the *Checks* tab, if needed.

> **i Note**
>
> For more information about the *Checks* tab, see Checks [page 17].

6. Choose *Save* and then choose *Activate*.

Related Information

- For more information about the function configuration, see How to Configure the Simple Transfer Stucture Function.
- For more information about common aspects of SAP Profitability and Performance Management functions, see Concepts for Key Users [page 6].
- For more information on how to add and remove functions in SAP Profitability and Performance Management environments, see Function Hierarchy [page 63].

1.4.2.9.1.1 Example: Transfer Structure

This scenario shows how the "Transfer Structure" type transfers the values from the original fields to other fields based on the input tab formula, header settings and rules.

Input

Contract	Product	Value Date	Amount	Premium
C1	P10	2016-01-01	100	50
C2	P20	2016-01-07	200	250
C3	P3	2016-01-12	400	500
C1	P10	2016-01-16	50	300
C1	P10	2016-01-01	1,000	400

Contract	Product	Value Date	Amount	Premium
C2	P20	2016-02-10	150	700

This is the input data of the function, in which it will be enriched by the *Transfer Structure* function.

In the input tab, there is a formula where the amount is computed: Amount = Amount/2.5

The formula will then be executed first.

Contract	Product	Value Date	Amount	Premium
C1	P10	2016-01-01	40	50
C2	P20	2016-01-07	80	250
C3	P3	2016-01-12	160	500
C1	P10	2016-01-16	20	300
C1	P10	2016-01-01	400	400
C2	P20	2016-02-10	60	700

In the header portion, it is stated that characteristics are grouped. That is why in the input, the characteristics that are the same will be grouped. You can see that rows 1 and 5 are the same so they will be grouped. Then the key figures will be summed up.

Contract	Product	Value Date	Amount	Premium
C1	P10	2016-01-01	440	450
C2	P20	2016-01-07	80	250
C3	P3	2016-01-12	160	500
C1	P10	2016-01-16	20	300
C2	P20	2016-02-10	60	700

The aggregated characteristics with the key figures being totaled is already shown in row 1.

In our first rule, we have a selection of contract C1 and P10 (rows 1 and 4) and it has a source field in the action tab where the *Amount* field is selected.

In our second rule, we have a selection of contract C2 and P20 (rows 2 and 5) and it has a source field in the action tab where the *Amount* field is selected.

Contract	Product	Value Date	Amount	Premium
C1	P10	2016-01-16	20	300
C1	P10	2016-01-01	440	450
C2	P20	2016-01-07	80	250
C2	P20	2016-02-10	60	700

Here the rules will kick in and will perform the following formulas where it is mentioned that in the first rule the field *Premium 1* will now be equivalent to *Amount*.

In the second rule the field *Premium 2* will be equivalent to *Amount*.

Contract	Product	Value Date	Amount	Premium	Premium 1	Premium 2
C1	P10	2016-01-16	20	300	20	0
C1	P10	2016-01-01	440	450	440	0
C2	P20	2016-01-07	80	250	0	80
C2	P20	2016-02-10	60	700	0	60

In the header portion, it is stated that fields that will be retained will be the following: All fields except Selection & Action->Source Fields.

This means that fields within the selection and the source fields in the action will be excluded from the output (columns *Contract*, *Product* and *Amount* will be excluded in the output).

Final Output

Value Date	Premium	Premium 1	Premium 2
2016-01-16	300	20	0
2016-01-01	450	440	0
2016-01-07	250	0	80
2016-02-10	700	0	60

1.4.2.9.2 Transfer Structure Type: Reverse Transfer Structure

This method transfers columns to values, sometimes also called “unpivoting of data”. It is typically used to turn costing-based data into account-based data.

Procedure

Function Access

Follow the steps below to access the *Transfer Structure* function of type “Reverse Transfer Structure”:

1. In the client where SAP Profitability and Performance Management is installed, choose *SAP Menu* > *Profitability and Performance Management* > *Modeling* > *Start My Environments* .
2. The *Environment* screen appears in a separate browser window. Choose an existing environment and continue. Within the environment, you can set up the newly added function.

Function Configuration

Follow the steps below to configure the *Transfer Structure* function of type “Reverse Transfer Structure”:

1. In edit mode, configure the following required fields in the header. For more information about the header, see the *Key Features* section in Transfer Structure [page 282].
   - *Transfer Structure Type* = “Reverse Transfer Structure”

- *Result Handling*
- *Suppress initial Result*
- *Ensure Distinct Result*
- *Include original Input Data*
- *Result Model Table*
- *Retail Fields*
- *Aggregate Result*

2. Define the input function to be used on the *Input* tab.

> **i Note**
>
> For more information about the *Input* tab, see Input [page 11].

3. Define the fields to be used on the *Signature* tab.

> **i Note**
>
> For more information about the *Signature* tab, see Signature [page 13].

4. On the *Rules* tab, each transfer structure rule defines semantically an if-then statement. The “if”-part selects which subset of the input data the rule is relevant to. The “then”-part is an action, and contains a list of fields and values that have to be assigned.
   Proceed as follows to configure the rules:
   1. On the *Rule* tab of your *Transfer Structure* function, choose + (*Add*).
   2. The *Add Details* screen is displayed. Enter the following information:
      - *Rule*
      - *Description*
   3. Choose *OK*.
   4. Select the created rule.
   5. In the *Selection* section of the created rule, choose + (*Add*).
   6. On the *Fields* screen that is displayed next, choose the field to be used and then choose *OK*.
   7. Select the field and choose *Selection Condition*.
   8. On the *Selection Condition* screen, define the selection condition to be used and choose *OK*.
   9. In the *Action* section of the created rule, choose + (*Add*).
   10. On the *Fields* screen that is displayed next, choose the field to be used and choose *OK*.
   11. Select the field and choose *Formula*.
   12. On the *Formula* screen that is displayed next, define the formula to be used and choose *OK*.
5. Define the checks to be used on the *Checks* tab, if needed.

> **i Note**
>
> For more information about the *Checks* tab, see Checks [page 17].

6. Choose *Save* and then choose *Activate*.

Related Information

- For more information about common aspects of SAP Profitability and Performance Management functions, see Concepts for Key Users [page 6].
- For more information on how to add and remove functions in SAP Profitability and Performance Management environments, see Function Hierarchy [page 63].

1.4.2.9.2.1 Example: Reverse Transfer Structure

This scenario shows how the "Reverse Transfer Structure" type transfers the values from the original fields to new fields based on the header settings and rule types.

Input

Product	Customer	Premium 1	Premium 2
PROD01	CUST01	10	0
PROD02	CUST02	0	20

In our rule types, it is mentioned that there would be additional fields in where we will transfer the field values from the original one to the new fields:

Product	Customer	Premium 1	Premium 2	Premium Type	Premium
PROD01	CUST01	10	0		
PROD02	CUST02	0	20		

In the first rule it is indicated that in the line which contains PROD01 and CUST01, the value of *Premium 1* would be transferred to *Premium*, while the value PREMIUM_1 would be entered in the field *Premium Type*:

Product	Customer	Premium 1	Premium 2	Premium Type	Premium
PROD01	CUST01	10	0	PREMIUM_1	10
PROD02	CUST02	0	20		

In the first rule it is indicated that in the line which contains PROD02 and CUST02, the value of *Premium 1* would be transferred to *Premium*, while the value PREMIUM_2 would be entered in the field *Premium Type*:

Product	Customer	Premium 1	Premium 2	Premium Type	Premium
PROD01	CUST01	10	0	PREMIUM_1	10
PROD02	CUST02	0	20	PREMIUM_2	20

In the header portion, it is stated that fields that will be retained will be the following: All fields except Selection & Action->Source Fields.

Final Output

Premium Type	Premium
PREMIUM_1	10
PREMIUM_2	20

1.4.2.10 Valuation

In financial accounting, valuation is the process of determining the values of financial instruments or other resources. The *Valuation* function provides a variety of different rule and line types to calculate financial or statistical measures, such as discounted value of money, as well as median, variance, and so on, which can be used in the process.

Valuation usually consists of several product-specific or service-specific steps. In SAP Profitability and Performance Management each rule or line of the valuation function represents one of these steps in the configuration. For example, for commercial insurance contracts discounting may be carried out at cash flow level and value aggregation used to calculate statistical properties or sum up discounted cash flows to present values of the observed contracts. Therefore, you can use hierarchical rules to trigger the sequential valuation steps.

Key Features

Rule Types

Characteristic Formula: Applies formulas and SAP HANA SQL functions that return a string to characteristics output field.

Discounting: Calculates discounted or (in investment finance) "present" value of future money, such as expected payments.

Duration: Calculates three different types of durations. Their outputs are of similar numerical value, but the theoretical concepts behind the calculations and the interpretation of results differ significantly, as do the practical applications.

Interpolation: Produces a new series by linear approximating values between (interpolation) and beyond (extrapolation) known-value time points. It does so by replacing user-specified initial values with linear approximated values. It is used in finance to determine interest rates for missing time points along the yield curve (i.e. term structure of the curve).

Key Figure Formula: Applies formulas and SAP HANA SQL functions that returns numeric values to key figure output field.

Line Item Valuations: As an umbrella for eight different line types, this rule type enables the following data manipulations:

- Output can be calculated separately for each change in selected granularity (*Balance Granularity Fields*)
- Input can be multiplied by a number, formula or selected field (*Factor*)
- Input can be filtered for certain *Selection* conditions.

Running Total: Like the line type "Running Balance" of rule type "Line Item Valuation", this rule type sequentially sums a selected field and updates this sum (aka "running total") for each row by adding it to the previous running total. Unlike the line type "Running Balance" of rule type "Line Item Valuation", this rule type takes the current row's value in the calculation of the running total. It is used in finance to calculate the amount of outstanding loan principal that forms the basis for interest calculation, for example.

Value Aggregation: Output can be calculated separately for each change in selected granularity and by limiting the range of rows from the current row (Lower / Upper Row Offset).

Rule Lines

Duration

- **Macaulay Duration** is the weighted average maturity of future cash flows of a financial instrument. The weight of each cash flow's maturity is determined by dividing the present value of the cash flow by the net present value of the observed instrument. Under the assumption of yearly compounding, Macaulay Duration is given by the following formula, where "CF_i" is the (absolute) amount of the i-th cash flow, T is the respective maturity of i-th cash flow, "r" is the yield of maturity of this financial instrument:

$$\frac{\sum T_i \times \frac{CF_i}{(1+r)^{T_i}}}{\sum \frac{CF_i}{(1+r)^{T_i}}}$$

- **Modified Duration** is a measure of price sensitivity, defined as the percentage derivative of price with respect to yield to maturity and compounding method. Under the same assumption modified duration is given by

$$\frac{\sum T_i \times \frac{CF_i}{(1+r)^{T_i}}}{\sum \frac{CF_i}{(1+r)^{T_i}}} \times \frac{1}{(1+r)}$$

- **Fisher-Weil Duration**: When rates from zero-coupon yield curve are used instead of the yield to maturity , the so-called Fisher-Weil duration is calculated with the same formula instead of Macaulay duration.

Interpolation

- **Extrapolation None**: Leading and trailing initial values are not extrapolated (replaced). For example, an input of [null, null, 1, 2, null, null, 5, null, null] returns [null, null, 1, 2, 3, 4, 5, null, null].
- **Extrapolation Linear**: Leading and trailing initial values are replaced by the values calculated by the linear line extended first or last two known values respectively.
- **Extrapolation Constant**: Extends the first and last known value to leading and trailing initial values respectively.

Line Item Valuations

- **Balance**: Returns the sum of an expression that is maintained in the *Value* input field of rows from the flow lookup function that matches the current row of input function by certain characters. The sum can then be multiplied with the returned value of an expression if this expression is maintained in input field *Factor*.
- **Formula**: Applies certain formulas and SQL functions to key figures and/or characteristics.
- **Lag**: Uses SAP HANA SQL *LAG* function to return the value of a previous row where the position is specified by the offset value in the same granularity set. The offset value should be positive, the default is "1". Its

widespread usage entails cases where preceding data needs to be compared with or applied to current data. For example, the previous period's interest rate needs to be applied to current period's interest payment calculation.

- **Lead**: Uses SAP HANA SQL *LEAD* function to return the value of a following row whose position is specified by the offset value in the same granularity set. The offset value should be positive, the default is "1". Its widespread usage entails cases where subsequent data needs to be compared with or applied to current data. For example, a running total of all future payments needs to be calculated on a data set containing outflows.
- **Register**: Just like the line type "Formula", this line type applies formulas and SQL functions to key figures and/or characteristics. Unlike the rule type "Formula", this line type does not have options "Flow Lookup Function" and "Lookup Result".
- **Running Balance** or **Running "Opening" Balance**: Similar to the rule type "Running Total", this rule type sums the values of a selected field of rows before the current row so that each following line's value is added to the previous running balance. Unlike the rule type "Running Total", this rule type considers the preceding row's value in the calculation – hence the term "opening" balance.
- **Scaled Weighted Average**: Calculates weighted mean by dividing the sum of weights times values with the sum of the weights, according to the following formula:

$$\frac{\sum_{i=1}^{n} W_i \times X_i}{\sum_{i=1}^{n} W_i}$$

- **Running Scaled Weighted Average**: Calculates a "running" weighted mean for rows before the current row by dividing the sum of weights times values with the sum of the weights.

Value Aggregation

- **Number of Rows**: Returns the number of rows of a selected field.
- **Minimum Value**: Returns the minimum value of a selected field.
- **Statistical Median**: Returns the median of a selected field.
- **Maximum Value**: Returns the maximum value of a selected field.
- **Sum Value**: Returns the sum of a selected field.
- **Arithmetical Mean**: Returns the average of a selected field.
- **Standard Deviation Square Root of Population Variance**: Returns standard deviation of a selected field by taking the square root of the population variance.
- **Standard Deviation Square Root of Sample Variance**: Returns standard deviation of a selected field by taking the square root of the sample variance.
- **Population Variance Value**: Returns the population variance of an expression as the sum of squares of the difference of <expression> from the mean of <expression>, divided by the number of rows remaining.
- **Sample Variance Value**: Returns the population variance of an expression as the sum of squares of the difference of <expression> from the mean of <expression>, divided by the number of rows remaining minus 1.

Procedure

Function Access

Follow the steps below to access the *Valuation* function:

1. In the client where SAP Profitability and Performance Management is installed, choose *SAP Menu* > *Profitability and Performance Management* > *Modeling* > *Start My Environments* .
2. The *Environment* screen appears in a separate browser window. Choose an existing environment and continue. Within the environment, you can set up the newly added function.

Function Configuration

Follow the steps below to configure the *Valuation* function:

1. In edit mode, configure the following required fields in the header:
   - *Result Handling*
   - *Suppress initial Result*
   - *Include original Input Data*
   - *Result Model Table*
2. Define the input function to be used on the *Input* tab.

   > **i Note**
   >
   > For more information about the *Input* tab, see Input [page 11].

3. Define the fields to be used on the *Signature* tab.

   > **i Note**
   >
   > For more information about the *Signature* tab, see Signature [page 13].

4. On the *Rules* tab, you can define the rule and line types of the *Valuation* function. For more information about the available rule types, see the *Key Features* section above.
   Proceed as follows to configure the rules:
   1. On the *Rule* tab of your *Valuation* function, choose + (*Add*).
   2. The *Add Details* screen is displayed. Enter the following information:
      - *Add Level*
      - *Rule*
      - *Rule Type*
      - *Description*
   3. Choose *OK*.
   4. Select the created rule.
   5. On the *Rule Lines* tab of the selected rule, choose + (*Add*).
   6. The *Add Details* screen is displayed. Enter the required information.
   7. Select the created rule line and enter the required information.

      > **i Note**
      >
      > Depending on the chosen rule type the rule line shows different fields.

5. Define the checks to be used on the *Checks* tab, if needed.

> **i Note**
>
> For more information about the *Checks* tab, see Checks [page 17].

6. Choose *Save* and then choose *Activate*.

Related Information

- For more information about common aspects of SAP Profitability and Performance Management functions, see Concepts for Key Users [page 6].
- For more information on how to add and remove functions in SAP Profitability and Performance Management environments, see Function Hierarchy [page 63].

1.4.2.10.1 Example: Running Total

This scenario shows the sequential summation capability of the *Valuation* function which updates the field by adding the previous running balance to the amount.

Input Data

Yield Curve Type	Period	Interpolated Yield	Amount
Yield Curve	0	–0.0001537	100.0000
Yield Curve	1	0.0043374	100.4337
Yield Curve	2	0.0088285	101.7735
Yield Curve	3	0.0096286	102.9165
Yield Curve	4	0.0104287	104.2372
Yield Curve	5	0.011203	105.7284
Yield Curve	6	0.0119945	107.4160

In order to aggregate the value, the system uses the following formula:

$$RT = \sum_{i}^{n} Principal_i$$

Based on the configuration, the key figure to be aggregated for the running total is “Amount”. The system computes the total of all the data and stores it in the additional column “Running Total”.

Result

Yield Curve Type	Period	Interpolated Yield	Amount	Running Total
Yield Curve	0	–0.0001537	100.0000	100.0000
Yield Curve	1	0.0043374	100.4337	200.4337
Yield Curve	2	0.0088285	101.7735	302.2072
Yield Curve	3	0.0096286	102.9165	405.1237
Yield Curve	4	0.0104287	104.2372	509.3609
Yield Curve	5	0.011203	105.7284	615.0893
Yield Curve	6	0.0119945	107.4160	722.5053

1.4.2.11 View

View is a data access function that can be used to select data and provide projections and aggregations on top of it. This view on the data can then be used for consumption in other functions like *Allocation*. In addition, a view has several options for fine-tuning data consumption. For example, it can use a table sample or fraction of the input data to select a specific time version of data in history tables or it can only provide input data if a run parameter precondition is met (for example, if `MY_READ_PARAM_FLAG` = "X"). A view can also run iterations of input function calls, including early exit checks.

Key Features

Header

In the header, you define the principal behavior of the view.

View Type:

- **Implicit Fields**: The view adopts all the fields from the input. Only the fields explicitly named in the output are used if you have defined an aggregation on the *Output* tab using field grouping.
- **Explicit Fields**: The view adopts only the fields explicitly named on the *Output* tab; all other fields are excluded.

Output

You can enter additional field details (such as select conditions, formulas, group aggregations and sort orders) on the *Output* tab. This is optional for the view type "Implicit Fields", but mandatory for the type "Explicit Fields".

Advanced

The following options are available on the *Advanced* tab:

- *Top*: You can enter a constant number or parameter in this field to restrict the data reading to a given absolute number of records. For example, Top = 100 reads only the first 100 records of the input.

- *Run Parameter Precondition*: If you enter a parameter condition here, the view only provides an output if this precondition is met. Otherwise, the output is 0 records.
- *Default Type*: If you choose “Default output, if input empty”, the view populates and displays results based on the assigned field value in the Output tab. This gives a modeler the option of producing one record table with self-assigned values.
- *Iteration Type*:
  The iteration types execute stored procedures of an input function in a loop when the *View* function is executed.
  If you choose the option “None”, no iteration takes place.
  For scenarios where an input function is a *Writer* function that writes data to a Model Table, you can use the following iteration types:
  - “For Loop”
  - “Reverse For Loop”

  There are scenarios where the input function contains a processing logic that is outside of the stored procedure, such as in a process chain. An example is the *Writer* function that writes data to BW. In this case, you can use the following iteration types:
  - “Application Server For Loop”
  - “Application Reverse For Loop”

  > **i Note**
  >
  > If you want the deletion to happen in every iteration when you write to model data, use “Application Server for Loop” and “Application Server Reverse For Loop”. This is only possible when the model writer type “Delete and Insert” is used.

- The system bases the number of loops it executes on the limits that you define in the following fields:
  - *Low*: Minimum number of loops to be executed
  - *High*: Maximum number of loops to be executed
- *Iteration Parameter*: In the *Iteration Parameter* field, you need to enter a parameter that contains the current loop number and thus makes it available for the input function as well.
- *Early Exit Check*: You can register an early exit check from the environment checks. This is applied to the view result and if the check is successful, the iteration is exited early.

  > **i Note**
  >
  > The *Early Exit Check* field is not available when selecting iteration type “Application Server For Loop” or “Application Server Reverse For Loop”.

- *Iteration Result*:
  - All Iterations: The result of all iterative calls of the view input is collected and provided as output.
  - Last Iteration: Only the output of the last iterative call is provided.

Procedure

Function Access

Follow the steps below to access the *View* function:

1. In the client where SAP Profitability and Performance Management is installed, choose *SAP Menu* > *Profitability and Performance Management* > *Modeling* > *Start My Environments* .
2. The *Environment* screen appears in a separate browser window. Choose an existing environment and continue. Within the environment, you can set up the newly added function.

Function Configuration

Follow the steps below to configure the *View* function:

1. In edit mode, configure the following required fields in the header. For more information about the header, see the *Key Features* section above.
   - *View Type*
   - *Suppress initial Result*
   - *Result Model Table*
2. Define the input function to be used on the *Input* tab.

   > **i Note**
   >
   > For more information about the *Input* tab, see Input [page 11].
3. Define the fields to be used on the *Signature* tab.

   > **i Note**
   >
   > For more information about the *Signature* tab, see Signature [page 13].
4. Define the fields to be used on the *Output* tab, if needed.

   > **i Note**
   >
   > For more information about the *Output* tab, see Output [page 15].
5. Define the checks to be used on the *Checks* tab, if needed.

   > **i Note**
   >
   > For more information about the *Checks* tab, see Checks [page 17].
6. Define the required fields on the *Advanced* tab, if needed:
   - *Top*
   - *Run Parameter Precondition*
   - *Default Type*
   - *Iteration Type*
   - *Low*
   - *High*
   - *Iteration Parameter*
   - *Early Exit Check*

- *Iteration Result*

> **i Note**
>
> For more information about the *Advanced* tab of the *View* function, see *Related Information*.

7. Choose *Save* and then choose *Activate*.

Related Information

- For more information about the *Advanced* tab of the *View* function, see View Function: Advanced Tab.
- For more information about common aspects of SAP Profitability and Performance Management functions, see Concepts for Key Users [page 6].
- For more information on how to add and remove functions in SAP Profitability and Performance Management environments, see Function Hierarchy [page 63].

1.4.2.11.1 Example: Aggregation

This example shows a simple aggregation of quantity and amount using the *View* function on the basis of a raw data table containing orders from different customers.

Input: VI - Order Table 1 (Raw Data)

Customer	Product	Quantity	Amount
CN002	PROD01	40	23
CN001	PROD02	20	20
CN004	PROD01	30	33
CN003	PROD02	30	25
CN001	PROD03	33	40
CN004	PROD03	25	28
CN002	PROD01	25	30

1. Group customers that have the same characteristics (from the field *Group*).

Customer	Product	Quantity	Amount
CN001	PROD01	20	20
	PROD02	33	40
CN002	PROD01	40	23
	PROD02	25	30
CN003	PROD03	30	25

Customer	Product	Quantity	Amount
CN004	PROD03	30	33
	PROD01	25	28

2. Count how many products there are in each grouped *Customer* (from the formula `COUNT(PRODUCT)`).

Customer	Product	Quantity	Amount
CN001	2 Products (PROD01 & PROD02)	20	20
		33	40
CN002	2 Products (PROD01 & PROD02)	40	23
		25	30
CN003	1 Product	30	25
CN004	2 Products (PROD03 & PROD01)	30	33
		25	28

3. Add up the *Quantity* per grouped *Customer* (from the formula `SUM(QUANTITY)`) and the *Amount* per grouped *Customer* (from the formula field `SUM(AMOUNT)`).

Customer	Product	Quantity	Amount
CN001	PROD01	53 = 20 + 33	60 = 20 + 40
	PROD02		
CN002	PROD01	65 = 25 + 40	53 = 23 + 30
	PROD02		
CN003	PROD03	30	25
CN004	PROD03	55 = 30 + 25	61 = 33 + 28
	PROD01		

4. Result: Aggregate View

Customer	Product	Quantity	Amount
CN001	2	53	60
CN002	2	65	53
CN003	1	30	25
CN004	2	55	61

1.4.2.11.2 Example: Data Selection Using Hierarchy

This example shows a simple hierarchy selection using the *View* function, in which the parent node is selected and its underlying child nodes are included in the filtered data.

You must specify a hierarchy name for the field that is used for selection on the environment level. Once the hierarchy name is assigned, you can view the hierarchy while you filter the data.

In this example, the environment field *Product* (ZE_PROD) is used and the hierarchy name PROD_PER_CNTRY is applied for this field.

We use the master data and hierarchy structure below:

Field	Value	Description	Is Node	Parent Value	Hierarchy Name
ZE_PROD	US	US	Yes		PROD_PER_CNTRY
ZE_PROD	PROD000001	PROD01	Default (No)	US	PROD_PER_CNTRY
ZE_PROD	PROD000002	PROD02	Default (No)	US	PROD_PER_CNTRY
ZE_PROD	PROD000003	PROD03	Default (No)	US	PROD_PER_CNTRY
ZE_PROD	PROD000004	PROD04	Default (No)	US	PROD_PER_CNTRY
ZE_PROD	PROD000005	PROD05	Default (No)	US	PROD_PER_CNTRY
ZE_PROD	DE	DE	Yes		PROD_PER_CNTRY
ZE_PROD	PROD000006	PROD06	Default (No)	DE	PROD_PER_CNTRY
ZE_PROD	PROD000007	PROD07	Default (No)	DE	PROD_PER_CNTRY
ZE_PROD	PROD000008	PROD08	Default (No)	DE	PROD_PER_CNTRY
ZE_PROD	PROD000009	PROD09	Default (No)	DE	PROD_PER_CNTRY
ZE_PROD	PROD000010	PROD10	Default (No)	DE	PROD_PER_CNTRY

Follow the steps below to define the hierarchy name:

1. Choose the *Environment* button.
2. Go to the *Environment Fields* tab.
3. Choose *Edit*.
4. Select the *Product* (ZE_PROD) field.
5. Enter the hierarchy name **PROD_PER_CNTRY**.

Once you have set up the hierarchy name, the hierarchy can be used as a selection condition.

Input Data: Product Table

Product (ZE_PROD)	Amount (ZE_AMT)	Currency Key (ZE_CURR)
PROD000001	5,000.00	USD
PROD000002	7,500.00	USD
PROD000003	85,000.00	USD
PROD000004	9,500.00	USD
PROD000005	5,200.00	USD
PROD000006	2,500.00	EUR
PROD000007	3,500.00	EUR
PROD000008	2,000.00	EUR
PROD000009	15,000.00	EUR

Product (ZE_PROD)	Amount (ZE_AMT)	Currency Key (ZE_CURR)
PROD000010	2,500.00	EUR

In the *View* function, you need to enter the *Product* (ZE_PROD) field:

1. Choose the *Edit* button.
2. On the *Input* tab, choose *Add*.
3. Select the *Product* (ZE_PROD) field, then choose *OK*. The field now appears on the *Input* tab.
4. Select the field and choose the *Selection Condition* button. The *Selection Condition* window is displayed.
5. In the search criteria, choose the F4 help button. The parent node of the hierarchy is displayed.
6. If you expand the nodes, the child values are displayed and can be selected as shown below:

   Product Hierarchy

Description
US
–→ PROD01
–→ PROD02
–→ PROD03
–→ PROD04
–→ PROD05
DE
–→ PROD06
–→ PROD07
–→ PROD08
–→ PROD09
–→ PROD10

7. Select the parent node with the description “DE”.

ZE_PROD (*Product*)	is	DE

8. The system now filters the results and displays the values under the parent node in the output of the *View* function.

Product (ZE_PROD)	Amount (ZE_AMT)	Currency Key (ZE_CURR)
PROD000006	2,500.00	EUR
PROD000007	3,500.00	EUR
PROD000008	2,000.00	EUR
PROD000009	15,000.00	EUR
PROD000010	2,500.00	EUR

1.4.2.11.3 Example: Loop

This scenario uses the looping feature of the *View* function. It calls the input function and runs multiple times.

Input: VI - Result Table

Input Number	Divisible by 3
1	NO

In a normal modeling scenario, the iteration is handled by calling a chain of functions ending with a writer. A simple scenario has been created to highlight the looping mechanism provided by the *View* function.

Prerequisite: The model table has one input record.

Initial Input Table

Input Number	Divisible
1	NO
2	NO
3	YES

1. The model table acts as a repository for both input (join) and output (writer), which will then be used in the loop.
   Assuming that there are already three records in the model table:
2. The Join acts as the process that must be run on the records before the result is written back to the model table. In this scenario, the join has been set up to perform the following three steps:
   1. Step 1: All records from the model table are collected and the maximum record is marked. Marking is done by performing `Iteration Counter = MAX (Input Number) OVER ()`, giving the result on the left.

Input Number	Dibisible by 3	Iteration Counter
3	YES	3

   2. Step 2: Select only the record that has the same iteration counter and input number. This can be configured by using complex selections tab: `WHERE Iteration Counter = Input Number`. In this case, the result of step 2 is on the left.

Input Number	Divisible by 3
4	NO

   3. Step 3:
      1. `ZQA_INPNO` is incremented by 1, the formula is Input Number = Input Number +1, so Input Number = 3+1.
      2. The record should be assessed if divisible by 3. The formula for this is as follows:

         ```
         DIVISIBLE = CASE WHEN MOD ((Input Number +1),3) = 0 THEN "YES"
         ELSE "NO"
         END
         ```

so

```
DIVISIBLE = CASE WHEN MOD ((3+1),3) = 0 THEN "YES"
ELSE "NO"
END
```

In this case, the condition statement provides a “NO” since 4 MOD 3 = 1.

3. The iteration counter must also be set to “Not Used” to prepare the writing to the model table. The result of step 3 is on the left.

Input Number	Divisible by 3
1	NO
2	NO
3	YES
4	NO

3. The final result from step 2 is inserted in the model table using a writer. If the model table is checked, it must have a result like the one on the left after the iteration.
   If the process needs to be run multiple times, it can be done via view.
4. A view with active looping functions must be used to call the writer multiple times. The main steps 1 to 3 are executed multiple times based on what has been defined on the *Advance* tab of the view.

> **i Note**
>
> If the iteration is set to loop from 1 to 50, the writer is called 51 times; once from the input tab of the view and 50 more times from the loop set (1-50). Do not assess the result using the view result, instead use the model table where you can see all the records that have been written since the last iteration.

1.4.3 Write and Adapter Functions

These functions can store data or hand over data to external systems for further processing. They also provide output to subsequent functions.

Functions belonging to this category include the following:

- File Adapter (Type "Export") [page 303]

  > **i Note**
  >
  > There is also a *File Adapter* function of type “Import” (see here [page 109]), which belongs to another function category (Information Functions [page 109]).

- Remote Function Adapter [page 305]
- Writer [page 323]

1.4.3.1 File Adapter (Type "Export")

The *File Adapter* function of type “Export” provides automated access to files so that results can be exported as file content.

Key Features

Header

In the header, you define the principal behavior of the *File Adapter* function. The following fields are available:

- *File IO Type*: Choose “Export” to export data into a server file.
- *File Format*: Refers to the definition of a file format, which is maintained centrally for the environment.
- *File Name*: Specifies the name of the file on the server that is used.
- *Number of Threads*: Multiple threads can reduce the export time. The maximum permitted value is 256.

Server Files

This is a helper tab that has no influence on the runtime of the function.

The *Refresh Directory List* button shows a list of files that are currently available on the server. The content of these files can also be viewed here. Use the *Select File* button to register it in the header as the file name to be used.

Small files can also be uploaded and downloaded but we recommend that you use server-side IT-driven mechanisms to manage files in the server directory.

Preview

This is a helper tab that has no influence on the runtime of the function.

Once the file name is set in the header, the *Preview* button allows you to preview the file.

Stage

This is a helper tab that has no influence on the runtime of the function.

Once the file name is set in the header, the *Stage* tab allows you to stage the file in a temporary table separating the data into columns. This makes it easier to analyze data, including filtering, sorting, and checking.

Mapping

The file columns can be mapped to existing fields in the environment. The *Field Mapping Proposal* button helps you to match columns to field names.

Optionally, formulas can be defined to convert data.

Procedure

> **Note**
>
> Before you can use the *File Adapter* function, you have to configure the *File Formats* section on the *Environment Details* screen. For more information about this section, see File Formats [page 60].

Function Access

Follow the steps below to access the *File Adapter* function of type "Export":

1. In the client where SAP Profitability and Performance Management is installed, choose *SAP Menu* > *Profitability and Performance Management* > *Modeling* > *Start My Environments*.
2. The *Environment* screen appears in a separate browser window. Choose an existing environment and continue. Within the environment, you can set up the newly added function.

Function Configuration

Follow the steps below to configure the *File Adapter* function of type "Export":

1. In edit mode, configure the following required fields in the header. For more information about the header, see the *Key Features* section above.
   - *File IO Type* = "Export"
   - *File Format*: Choose the file format configured in *Environment* > *File Formats*
   - *Number of Threads*
2. Define the input function to be used on the *Input* tab.

   > **Note**
   >
   > For more information about the *Input* tab, see Input [page 11].
3. Define the fields to be used on the *Signature* tab.

   > **Note**
   >
   > For more information about the *Signature* tab, see Signature [page 13].
4. Choose *Field Mapping Proposal* on the *Mapping* tab to generate the mapping of columns to field names with field attributes.
5. Define the checks to be used on the *Checks* tab, if needed.

   > **Note**
   >
   > For more information about the *Checks* tab, see Checks [page 17].
6. Choose *Save* and then choose *Activate*.

Related Information

- For more information about common aspects of SAP Profitability and Performance Management functions, see Concepts for Key Users [page 6].

- For more information on how to add and remove functions in SAP Profitability and Performance Management environments, see Function Hierarchy [page 63].

1.4.3.2 Remote Function Adapter

The *Remote Function Adapter* function provides automated communication capabilities to other applications and systems so that they can be included in calculations and processes.

Key Features

Header

The following Remote Function Adapter types are available:

- **Finance Account Payable**
  This function type enables you to post an accounts payable entry to SAP ERP or SAP S/4HANA. Each record is mapped to one posting, a GL account on one side and a vendor account on the other. If an RFC destination is defined at the environment level, the entries are posted in that system.
  For FI-GL integration scenarios with SAP ERP or SAP S/4HANA, this functionality will enable posting of general ledger documents from SAP Profitability and Performance Management. It makes use of the BAPI `BAPI_ACC_DOCUMENT_POST`. For further documentation refer to the function module documentation.
- **Finance Accounts Receivable**
  This function type enables you to post an accounts receivable entry to SAP ERP or SAP S/4HANA. Each record is mapped to one posting, a customer account on one side and a GL account on the other. If an RFC destination is defined at the environment level, the entries are posted in that system.
  For FI-GL integration scenarios with SAP ERP or SAP S/4HANA, this functionality will enable posting of general ledger documents from SAP Profitability and Performance Management. It makes use of the BAPI `BAPI_ACC_DOCUMENT_POST`. For further information refer to the function module documentation.
- **Finance Extended**
  This is a combination of the function types “Finance General Ledger” and “Finance Accounts Payable/ Receivable”. If an RFC destination is defined at the environment level, the entries are posted in that system. The posting is only completed when one of the following GL accounts is mapped:
  - `accountgl[1]`
    In this case, the system creates a receivable document.
  - `accountgl[[2]`
    In this case, the system posts a payable document.

  > **i Note**
  >
  > The posting is not completed when both accounts `accountgl[1]` and `accountgl[2]` are mapped. In such cases, the system displays an error during the function run. Therefore, we recommend you map only one GL account.

  For FI-GL integration scenarios with SAP ERP or SAP S/4HANA, this functionality will enable posting of general ledger documents from SAP Profitability and Performance Management. It makes use of the BAPI `BAPI_ACC_DOCUMENT_POST`. For further information refer to the function module documentation.

- **Finance General Ledger**
  This function type enables you to post a general ledger entry to SAP ERP or SAP S/4HANA. Each record is mapped to one posting, a debit and a credit on each side. If an RFC destination is defined at the environment level, the entries are posted in that system.
  For FI-GL integration scenarios with SAP ERP or SAP S/4HANA, this functionality will enable posting of general ledger documents from SAP Profitability and Performance Management. It makes use of the BAPI `BAPI_ACC_DOCUMENT_POST`. For further information refer to the function module documentation.
  For more information about this function type, see RFA Type: Finance General Ledger [page 311].
- **Finance General Ledger Items**
  This function type enables you to post a general ledger entry to SAP ERP or SAP S/4HANA, with multiple lines per document. A grouping field determines which lines are to be grouped into one document. Debit amounts (positive) and credit amounts (negative) should add up to zero.
  For the Remote Function Adapter types "Finance General Ledger" and "Finance General Ledger Items", you can maintain the following values and perform the respective postings using the *Document Status* field:
  - header-docu_status = "2": The system only simulates the document posting without posting the document in the target system.
  - header-docu_status = <blank>: The system posts a normal document.
  - header-docu_status = "V": The system posts a parked document.

  The function type uses the BAPI `BAPI_ACC_DOCUMENT_POST`. For more information, see the function module documentation.
- **HANA R Script**
  An external R script procedure is called. The expected interface of the external R script is displayed on the *Rules* tab. The R script can be entered directly on the *Rules* tab.
- **HANA Stored Procedure**
  An external SAP HANA stored procedure is called. The expected interface of the external SAP HANA stored procedure is displayed on the *Rules* tab. You need to define the authoring schema that the system uses to store the procedure together with the procedure name.
  For more information about this function type, see Example of a SAP HANA Stored Procedure [page 320].
- **External Function**
  The specified (remote) function module is called during the run. If the RFC destination specifies the local system, the function module can be either a regular function module or a remote-enabled module. If the RFC destination specifies a system other than the local system, the assigned function module has to be a remote-enabled module. The expected local interface of the external function is displayed on the *Rules* tab.
- **Replicate CO Master Data**
  Replicates master data and hierarchies for specific and main controlling module fields like *Cost Center*, *Cost Element*, *Profit Center* etc. from an integrated SAP S/4HANA or SAP ERP system into SAP Profitability and Performance Management environment fields.
- **Sales and Distribution**
  For integration scenarios with SAP ERP or SAP S/4HANA, this functionality enables creation of sales order, inquiry or quotation in SAP ERP or SAP S/4HANA Sales and Distribution module from SAP Profitability and Performance Management.
  It is mass processing enabled. Multiple and different documents can be created at once.
  It makes use of the Sales and Distribution BAPIs `BAPI_SALESORDER_CREATEFROMDAT2`, `BAPI_INQUIRY_CREATEFROMDATA2` and `BAPI_QUOTATION_CREATEFROMDATA2`.
- **Financial Statement Items**
  For FI-GL integration scenarios with SAP S/4HANA, this functionality enables retrieving of one or multiple (trial) balances from FI into SAP Profitability and Performance Management. This adapter leverages the `FAC_FINANCIAL_STATEMENT_SRV` SAP OData Service on the FI side.

The following header fields are available only for Remote Function Adapter type “Financial Statement Items”:

- *Decimal Separator*: Defines the separator to be used within the function.
- *HTTP RFC Destination*: Here you can specify the RFC destination type “H” of the remote system, where input data is stored.

- **Purchase Order**
  For integration scenarios with SAP ERP or SAP S/4HANA, this new Remote Function Adapter type allows you to create a purchase order in Material Management from SAP Profitability and Performance Management. It uses the BAPI `BAPI_PO_CREATE1`. For more information, see the function application help and function module documentation.
- **Secondary Cost Elements Posting**
  For integration scenarios with SAP ERP or SAP S/4HANA, this new Remote Function Adapter type allows you to post manual cost allocation with secondary cost elements posting in Controlling from SAP Profitability and Performance Management. It uses the BAPI `BAPI_ACC_MANUAL_ALLOC_POST`. For more information, see the function application help and function module documentation.
- **Primary Cost Reposting**
  For integration scenarios with SAP ERP or SAP S/4HANA, this Remote Function Adapter type allows you to repost primary cost posting in Controlling from SAP Profitability and Performance Management. It uses the BAPI `BAPI_ACC_PRIMARY_COSTS_POST`.
- **Replicate InfoObject Master Data**
  This functionality enables to replicate the master data of info objects from remote systems (for example SAP ERP or SAP S/4HANA) into corresponding environment fields within SAP Profitability and Performance Management.
  For more information about this function type, see RFA Type: Replicate InfoObject Master Data [page 318].
- **Post Journal Entries for Group Reporting**
  This Remote Function Adapter type enables you to post and simulate journal entries in group reporting in SAP S/4HANA using the synchronous inbound service `API_CNSLDTNGRPJRNLENTR`.
  For more information about this function type, see RFA Type: Post Journal Entries for Group Reporting [page 314].

Depending on the selected Remote Function Adapter type, a subset of the following fields is available:

- *RFC Destination*
  The RFC destination of type “ABAP Connection” specified in this field is used to post documents, call external reports, get master data and hierarchies. The RFC destination must exist in SM59 before use. If no value is entered in this field, the local client and system is used.
- *HTTP RFC Destination*
  The RFC destination of type “HTTP Connection” specified in this field is used if you need to execute SAP OData services (for example, for the Remote Function Adapter type “Financial Statements Items”).
- *Field Length Extension Mode*
  This read-only field indicates the data type of the amount fields that the remote system specified in the RFC destination uses:
  - Not extended, compatibility mode – amount values of type DECIMAL(23,4)
  - Target mode – amount values of type DECIMAL(31,8)
- *Authoring Schema*
  This is mapped in the physical schema into transported or stored objects using the SAP HANA database.

> **Note**
>
> You need to map the physical schema that the stored procedure is located in to the authoring schema. For more information, see *Maintain Schema Mapping for NXI Schema* (part of the Administration Guide for SAP Profitability and Performance Management).

- *Stored Procedure*
  This is the name of the stored procedure to be executed.
- *Decimal Separator*
  Defines the separator to be used within the function.
- *Function Name*
  Specifies the name of a remote NetWeaver Function or a web service which is called to display the expected interface on the *Rules* tab.

All other header options, such as *Include Original Input Data*, *Result Handling*, *Suppress Initial Results* and *Result Model Table* are referred to as "Functional Building Blocks" and are not specific to the Remote Function Adaper function. For more information about these options, see Header [page 10].

Input

For more information about modeling input, see Input [page 11].

Rules

- Rule format: Table Form
  This is available for the following remote function types: Finance, Replicate CO Master Data and Sales Distribution
  - Component
    Corresponding component item relevant for the specific Remote Function Adapter
  - Name
    Component description
  - Declaration Type
    Input: Entered field or value that is sent to the corresponding component in the target system defined in the RFC destination. If the RFC destination is not defined, the input will be posted or updated within the source system.
    Output: The field specified here receives the corresponding component value from the target system defined in the RFC destination, and is included in the run result.
  - Is Mandatory
    Green: *Is Mandatory* field is set to "Yes". This indicates that an entry has to be made for this component.
    Orange: *Is Mandatory* field is set to "Yes". This indicates that an entry has not been made for this component.
    Grey: *Is Mandatory* field is set to "No". This indicates that an entry is not required for this component.
  - Field
    Fields that you can use with this function based on the input function.
  - Description
    Field description

> **Note**
>
> To check which fields have been defined for each function type, proceed as follows:

1. Go to transaction **/n/nxi/p1_mf**
2. From here go to *Rule Types* > *Remote Function Adapter Mapping*.
3. The system gives you the option to select an RFA type (for example, Finance General Ledger).
4. This provides you with an overview of the fields defined for the specific function.
5. You can make certain changes to the mapping list. Open a customer incident to report a change request.

- Rule format: Template and Statement
  This is available for the following remote function types: HANA Stored Procedure and External Function
  - Template
    Is used as the template guide for creating the procedure that is called in the Remote Function Adapter – HANA Stored Procedure and External Function.
  - Statement
    The system displays the expected interface, provided you have defined the procedure in the header of the function.

  This is available for remote function type: HANA R Script
  - Template
    Is used as the template guide for creating scripts that are executed by the Remote Function Adapter – HANA R Script.
  - Statement
    You can enter script directly on the user interface of the Remote Function Adapter.

RFC Destination

For FI-GL and FI-GLI types, there are two ways of adding the RFC destination:

- Environment level
- Function level

If the RFC destination is maintained on both an environment level and function level, the RFC destination to be followed gets maintained on the function level.

> **Example**
>
> Given the following setup:
>
> 1. Environment Level – RFC Destination: ABC
> 2. RFA – FI-GLI – RFC Destination: DEF
>
> the RFA function will post the document in the DEF system.

For SAP Profitability and Performance Management 3.0 SP07 or below, you cannot maintain an RFC destination inside these function types, but you can for SP08 and above. See SAP Note 2902513 for further details.

Procedure

Function Access

Follow the steps below to access the *Remote Function Adapter* function:

1. In the client where SAP Profitability and Performance Management is installed, choose *SAP Menu* > *Profitability and Performance Management* > *Modeling* > *Start My Environments*.
2. The *Environment* screen appears in a separate browser window. Choose an existing environment and continue. Within the environment, you can set up the newly added function.

Function Configuration

Follow the steps below to configure the *Remote Function Adapter* function:

1. In edit mode, configure the following required fields in the header. For more information about the header, see the *Key Features* section above.
   - *Remote Function Adapter Type*
   - *RFC Destination*
   - *Function Name*
   - *Field Length Extensions Made*
   - *HTTP RFC Destination*
   - *Decimal Separator*
   - *Suppress Initial Results*
   - *Result Model Table*
   - *Include Original Input Data*
   - *Result Handling*
   - *Authoring Schema*
   - *Stored Procedure*

   > **i Note**
   >
   > Depending on the chosen Remote Function Adapter type the header shows different fields.

2. Define the input function to be used on the *Input* tab.

   > **i Note**
   >
   > For more information about the *Input* tab, see Input [page 11].

3. Define the fields to be used on the *Signature* tab.

   > **i Note**
   >
   > For more information about the *Signature* tab, see Signature [page 13].

4. Define rules on the *Rules* tab.

   > **i Note**
   >
   > Depending on the chosen function type the *Rule* tab shows different rule configurations.

5. Define the checks to be used on the *Checks* tab, if needed.

> **i** Note
>
> For more information about the *Checks* tab, see Checks [page 17].

6. Choose *Save* and then choose *Activate*.

Related Information

- For more information about common aspects of SAP Profitability and Performance Management functions, see Concepts for Key Users [page 6].
- For more information on how to add and remove functions in SAP Profitability and Performance Management environments, see Function Hierarchy [page 63].

1.4.3.2.1 RFA Type: Finance General Ledger

The Remote Function Adapter type “Finance General Ledger” enables you to post a general ledger entry to SAP ERP or SAP S/4HANA. Every record is mapped to one entry containing a debit posting and a credit posting. If an RFC destination is defined at environment level, the entries are posted in that system.

For FI-GL integration scenarios with SAP ERP or SAP S/4HANA, this function enables the posting of general ledger documents from SAP Profitability and Performance Management. It uses the BAPI `BAPI_ACC_DOCUMENT_POST`.

Procedure

Preparation

Consider the following points before you configure the RFA type “Finance General Ledger”:

1. **Database Connection**
   Specify the database connection “DBCON” at environment level.
2. **RFC Destination**
   The RFC destination can be configured either at environment level or at function level by assigning a value or field for the component `control-rfcdest`.

   > **i** Note
   >
   > If you do not specify a RFC destination, the system posts to the local client and system.

3. **Input Function**
   If you want to post general ledger items to SAP ERP or SAP S/4HANA, the remote function requires an input function that maps the fields from the input function to the components on the *Rules* tab.

> **Note**
>
> RFA type “Finance General Ledger” uses the standard accounting interface (BAPI). This means that the input records are validated according to the standard SAP Financial Accounting rules.

The input function can be a Model Table, Model View, or the result of another function, provided it contains the necessary fields.

> **Example**

Posting Date (ZE_PODAT)	Company (ZE_FCMP)	GL Account (ZE_FGLC)	Cost Center (ZE_COSCENT)	Sender Cost Center (ZE_SCNTR)	Item Text (ZE_ITMTXT)	Debit Amount (ZE_DEBITAMT)	Currency Key (ZE_CUKY)	Credit Amount (ZE_CREDITAMT)	Transaction Amount (ZE_TRANSAMT)	Distribution Rate (ZE_DISTRATE)	Object Key (ZE_OBJKEY)
20190101	0001	0000405200	0000009200	0000009400	Admin Cost	95000	EUR	-95000	495000	19	

> **Note**
>
> The entries used (for example for *Company*, *GL Account*, *Cost Center*, and so on) are for the purposes of this example only.

Configuration

Configure the RFA function as follows:

1. In edit mode, choose the *Add* button.
2. The function window appears. Make entries in the required fields:
   - *Function*
   - *Description*
   - *Function Type* = “Remote Function Adapter”
3. If necessary, make entries in the optional fields:
   - *Event Handling*
   - *Processing Type*
   - *Partitioning*
4. Choose *Ok* and save your changes.
5. In edit mode, enter the following function details:
   1. *Remote Function Adapter Type* = “Finance General Ledger”
   2. *RFC Destination*: If this is not set, the system uses the RFC destination defined in the *Advanced* tab of the environment. If you want to replicate the master data from a remote system, you can define the RFC destination here. If there are no RFC destinations assigned the system uses the local client and system.
   3. On the *Input* tab, specify the input function.

6. On the *Rules* tab, set the following mapping:

Component	Name	Function Type	Is Mandatory	Field
header-obj_key	Reference Key	FI-GL	X	ZE_OBJKEY
header-comp_code	Company Code	FI-GL	X	ZE_FGLC
header-doc_date	Document Date in Document	FI-GL	X	ZE_PODAT
header-doc_type	Document Type	FI-GL	X	'SA'
header-header_txt	Document Header Text	FI-GL		COST ALLOCATION - FIGL'
header-pstng_date	Posting Date in the Document	FI-GL	X	ZE_PODAT
header-username	User Name	FI-GL	X	SY-UNAME
accountgl[1]-gl_account	General Ledger Account (Debit)	FI-GL	X	ZE_FGLC
accountgl[1]-item_text	Item Text (Debit)	FI-GL		ZE_ITMTXT
accountgl[1]-cost-center	Cost Center (Debit)	FI-GL		ZE_COSCENT
accountgl[2]-gl_account	General Ledger Account (Credit)	FI-GL	X	ZE_FGLC
accountgl[2]-item_text	Item Text (Credit)	FI-GL		ZE_ITMTXT
accountgl[2]-cost-center	Cost Center (Credit)	FI-GL		ZE_SCNTR
amount[1]-currency	Currency Key (Debit)	FI-GL	X	ZE_CUKY
amount[1]-amt_doccur	Amount in Document Currency (Debit)	FI-GL	X	ZE_DEBITAMT
amount[2]-currency	Currency Key (Credit)	FI-GL	X	ZE_CUKY
amount[2]-amt_doccur	Amount in Document Currency (Credit)	FI-GL	X	ZE_CREDITAMT

7. Once you have completed the mapping, choose *Activate*.
8. Choose *Run*.
   The result is the same as the input and the configuration entered on the *Rules* tab. All of the mapped records are posted to the SAP S/4HANA system.
   The document number is then entered in the *Object Key* field.

Example

Posting Date (ZE_PODAT)	Company (ZE_FCMP)	GL Account (ZE_FGLC)	Cost Center (ZE_COSCENT)	Sender Cost Center (ZE_SCNTR)	Item Text (ZE_ITMTXT)	Debit Amount (ZE_DEBITAMT)	Currency Key (ZE_CUKY)	Credit Amount (ZE_CREDITAMT)	Transaction Amount (ZE_TRANSAMT)	Distribution Rate (ZE_DISTRATE)	Object Key (ZE_OBJKEY)
20190101	0001	0000405200	0000009200	0000009400	Admin Cost	95000	EUR	-95000	495000	19	**0100215405000012019**

The object key 0100215405000012019 is comprised as follows: document number + company code + fiscal year:

- Document Number = 0100215405
- Company Code = 0001
- Fiscal Year = 2019

9. Check if the entries have been posted successfully:
    1. Open the target system by launching the transaction FBL3N in the remote system or in the SAP Easy Access menu, under *Accounting* > *Financial Accounting* > *General Ledger* > *Document* > *Display*.
    2. Enter the following details:
        - *Document Number*
        - *Company Code*
        - *Fiscal Year*
    3. Choose *Continue*.
    4. The document is displayed in the SAP S/4HANA system.

1.4.3.2.2 RFA Type: Post Journal Entries for Group Reporting

The Remote Function Adapter type “Post Journal Entries for Group Reporting” enables you to post and simulate journal entries in group reporting in SAP S/4HANA using the synchronous inbound service API_CNSLDTNGRPJRNLENTR.

The service allows you to manually post journal entries to adjust financial reports, standardize entries, and consolidate entries according to group requirements. You can specify the relevant consolidation group, consolidation unit or consolidation unit pair as well as the local currency or group currency amounts for the journal entry line items according to the version, fiscal year, posting period, and document type.

This service provided by SAP S/4HANA Group reporting is published on the SAP API Business Hub and is available only from S/4HANA 2022 onwards. For more information about this API, see SAP API Business Hub or APIs for Group Reporting.

Procedure

Preparation

Consider the following before you configure this RFA type:

1. **Database Connection**
   Specify the database connection (“DBCON” or “DEFAULT”) at environment level.
2. **HTTP RFC Destination**
   Specify the RFC destination type “H” of the target Group Reporting system at function level.
3. **Input Function**
   If you want to post journal entries in group reporting in SAP S/4HANA, the remote function requires an input function that maps the fields from the input function to the components on the *Rules* tab.
   The input function can be a Model Table, a Model View or the result of another function, provided it contains the necessary fields.

Configuration

Configure the RFA function as follows:

1. In edit mode, choose *Add*.
2. The function window appears. Make entries in the required fields:
   - *Function*
   - *Description*
   - *Function Type* = “Remote Function Adapter”

   If necessary, specify the values for the optional fields:
   - *Event Handling*
   - *Processing Type*
   - *Partitioning*
3. Choose *Ok* and save your changes.
4. In edit mode, enter the following function details:
   - *Remote Function Adapter Type* = “Post Journal Entries for Group Reporting”
   - *HTTP RFC Destination*: Maintained RFC connection at “HTTP Connections to ABAP Systems” type.
   - On the *Input* tab, specify the input function.
5. On the *Rules* tab, set the following mapping:

Component	Name	Declaration Type	Is Mandatory	Field
control-grouping	Document Grouping by	Input	Yes	`ZE_BASEUNIT`
control-ordering	Document Order by	Input	No	
contorl-sim_fg	Simulation	Input	No	''
gr-FiscalYear	Fiscal Year	Input	Yes	`ZE_FISCALYR`
gr-ConsolidationVersion	Consolidation Version	Input	Yes	`ZE_CONVERSION`
gr-FiscalPeriod	Fiscal Period	Input	Yes	`ZE_FISCALPRD`

Component	Name	Declaration Type	Is Mandatory	Field
gr-ConsolidationChartofAccounts	Consolidation Chart of Accounts	Input	Yes	ZE_CONSCHARTOFACCT
gr-ConsolidationDocumentType	Consolidation Document Type	Input	Yes	ZE_CONSDOCTYPE
gr-DocumentItemText	Document Item Text	Input	No	
gr-ConsolidationUnit	Consolidation Unit	Input	Yes	ZE_CONSUNIT
gr-ConsolidationGroup	Consolidation Group	Input	No	
gr-ConsolidationUnit1	Consolidation Unit 1	Input	No	
gr-ConsolidationUnit2	Consolidation Unit 2	Input	No	
gr_item-ConsolidationPostingItem	Consolidation Posting Item	Input	Yes	ZE_CONSPOSTITEM
gr_item-FinancialStateItem	Financial State Item	Input	Yes	ZE_FINSTATEITM
gr-item-ConsolidationVersion	Consolidation Version	Input	Yes	ZE_CONVERSION
gr_item-SubItem	Subitem	Input	Yes	ZE_SUBITEM
gr_item-AmountInLocalCurrency	Amount in Local Currency	Input	No	ZE_AMTLOCALCURR
gr_item-LocalCurrency	Local Currency	Input	No	ZE_LOCALCURR
gr_item-AmountInGroupCurrency	Amount in Group Currency	Input	No	
gr_item-GroupCurrency	gr_item-GroupCurrency	Input	No	
gr_item-AmountInTransactionCurrency	Amount in Transaction Currency	Input	No	
gr_item-TransactionCurrency	Transaction Currency	Input	No	
gr_res-ConsolidationDocumentNumber	Document Number of an Accounting Document	Input	Yes	ZE_CONSDOCNUM
gr_res-ConsolidationVersion	Consolidation Version	Input	No	
gr_res-FiscalPeriod	Posting Period	Input	No	
gr_res-ConsolidationChartOfAccounts	Consolidation Chart of Accounts	Input	No	
gr_res-PostingLevel	Posting Level	Input	No	

6. Once the mapping is complete, choose *Activate*.
7. Run the function.

8. The result is the same as the input and the configuration entered on the *Rules* tab. All mapped records are posted to the SAP S/4HANA system.
   The document number returned by successful posting then appears in the *Consolidation Document Number* field.

Example

Input

Fiscal Year (ZE_FISCALYR)	Consolidation Version (ZE_CONVERSION)	Fiscal Period (ZE_FISCALPRD)	Consolidation Chart of Accounts (ZE_CONSCHARTOFACCT)	Consolidation Document Type (ZE_CONSDOCTYPE)	Document Item Text (ZE_DOCITMTXT)	Consolidation Unit (ZE_CONSUNIT)	Consolidation Posting Item (ZE_CONSPOSTITEM)	Financial Statement Item (ZE_FINSTATEITM)	Subitem (ZE_SUBITEM)	Amount in Local Currency (ZE_AMTLOCALCURR)	Local Currency (ZE_LOCALCURR)
2021	B10	012	Y1	B1	PAPM GROUP REPORTING ADAPTER	1000	1	111100	900	10.00	USD
2021	B10	012	Y1	B1	PAPM GROUP REPORTING ADAPTER	1000	2	111100	915	–10.00	USD

Output

Fiscal Year (ZE_FISCALYR)	Consolidation Version (ZE_CONVERSION)	Fiscal Period (ZE_FISCALPRD)	Consolidation Chart of Accounts (ZE_CONSCHARTOFACCT)	Consolidation Document Type (ZE_CONSDOCTYPE)	Document Item Text (ZE_DOCITMTXT)	Consolidation Unit (ZE_CONSUNIT)	Consolidation Posting Item (ZE_CONSPOSTITEM)	Financial Statement Item (ZE_FINSTATEITM)	Subitem (ZE_SUBITEM)	Amount in Local Currency (ZE_AMTLOCALCURR)	Local Currency (ZE_LOCALCURR)	Consolidation Document Number (ZE_CONSDOCNUM)
2021	B10	012	Y1	B1	PAPM GROUP REPORTING ADAPTER	1000	1	111100	900	10.00	USD	1000000124

Fiscal Year (ZE_FISCALYR)	Consolidation Version (ZE_CONVERSION)	Fiscal Period (ZE_FISCALPRD)	Consolidation Chart of Accounts (ZE_CONSCHARTOFACCT)	Consolidation Document Type (ZE_CONSDOCTYPE)	Document Item Text (ZE_DOCITMTXT)	Consolidation Unit (ZE_CONSUNIT)	Consolidation Posting Item (ZE_CONSPOSTITEM)	Financial Statement Item (ZE_FINSTATEITM)	Subitem (ZE_SUBITEM)	Amount in Local Currency (ZE_AMTLOCALCURR)	Local Currency (ZE_LOCALCURR)	Consolidation Document Number (ZE_CONSDOCNUM)
2021	B10	012	Y1	B1	PAPM GROUP REPORTING ADAPTER	1000	2	111100	915	–10.00	USD	1000000124

Related Information

For more information about the RFA type “Post Journal Entries for Group Reporting”, see Introduce the new RFA Type – Post Journal Entries for Group Reporting and how to configure it in SAP PaPM Application.

1.4.3.2.3 RFA Type: Replicate InfoObject Master Data

The Remote Function Adapter (RFA) type “Replicate InfoObject Master Data” enables you to replicate the master data and hierarchy of the InfoObjects from remote systems (for example, SAP ERP or SAP S/4HANA) into the corresponding environment fields within SAP Profitability and Performance Management.

Procedure

To replicate the data from the InfoObject, a model table is used as an input of the RFA function.

You need to create an environment field because only environment fields can be used to replicate InfoObject master data.

Input Data: Model Table

ENV_FLD_FR	ENV_FLD_TO
IO_SOURCE	ENV_FLD_RESULT

Explanation of the fields in the above table:

- `ENV_FLD_FR`: An environment field used as a column header in the model table which contains the InfoObject.
- `ENV_FLD_TO`: An environment field used as a column header in the model table which holds the target field.
- `ENV_FLD_RESULT`: An environment field which holds the replicated master data.
- `IO_SOURCE`: An InfoObject field used to replicate the master data.

> **i Note**
>
> The input function for the RFA can be a model table, model view or result of other functions as long as it contains the necessary fields.

To configure the RFA function, follow the steps below:

1. In edit mode, choose the *Add* button.
2. The *Function* window appears. Fill in the required fields:
   - *Function*
   - *Description*
   - *Function Type* = "Remote Function Adapter"
3. Fill in optional fields as necessary:
   - *Event Handling*
   - *Logging*
   - *Processing Type*
   - *Partitioning*
4. Choose *Ok* and save your changes.
5. In edit mode, fill in the following function details:
   1. *Remote Function Adapter Type* = "Replicate InfoObject Master Data"
   2. *RFC Destination*: If you want to replicate the master data from a remote system, you can define the RFC destination here.
      If you do not enter anything here, the system uses the RFC destination defined in the *Advanced* tab of the environment.
      If there are no RFC destinations assigned at all, the local client and system is used.
   3. On the *Input* tab, specify the input function. Use the model table as described in the *Input Data* table above.
   4. Set up the rules. The following two components are mandatory:
      - *infoobject-field*: The field assigned to this component is an environment field that holds the replicated master data.
        Based on the input data, the field that you need to assign is `ENV_FLD_RESULT`.
      - *infoobject-infoobject*: The field assigned to this component is the InfoObject field that is the source of master data to be replicated.
        Based on the input data, the field that you need to assign is `IO_SOURCE`.
6. Save and activate the function.
7. After executing the function, check whether the master data has been replicated.
   To do this, proceed as follows:
   1. Go to *Environment* > *Environment Fields* and search for the field `ENV_FLD_RESULT`.
   2. Select the field and choose *Open Master Data and Hierarchy*.

3. The *Data Editor* screen appears and displays all the master data and hierarchies that have been replicated.

1.4.3.2.4 Example of a SAP HANA Stored Procedure

Prerequisite

The remote function adapter requires input data which is then processed by the SAP HANA Stored Procedure.

In this example, simple input data is used in the form of a model table, to show the connection and logic of the remote function adapter SAP HANA stored procedure type.

1. Create a model table.
2. Name it “RFA Input”, for example.
3. On the right-hand side of the screen, configure the model table by entering the following:
   - *Model Table Source*: Environment
   - *Transport Data*: Default (No)
4. Add a field.

   > **Example**
   >
   > In this example, the field `JB_PROD` (characteristic, length = 50) has been added to show the relationship between input data and the remote function adapter.

5. Choose *Maintain Data* and add an entry.

   > **Example**
   >
   > In this example, the added data is `PRD01`.
   >
   > | Product field (JB_PROD) |
   > | --- |
   > | PRD01 |

6. Now the input has been defined and ready to be used.

Remote Function Adapter

The *Remote Function Adapter* function provides automated communication capabilities to other applications and systems so that they can be included in calculations and processes.

One of these is **`HANA STORED PROCEDURE`**, where an external SAP HANA stored procedure can be called by the function to process the input function in step 6. .

How to set up the remote function adapter

1. Create a remote function adapter.
2. Name it “RFA Function”, for example.
3. On the right-hand side of the screen, configure the header of the remote function adapter by making the following entries:
   - *Function Type*: **HANA STORED PROCEDURE**
   - *Supress Initial Result*: Default
   - *Result Model Table*: Empty (by default)
   - *Authoring Schema*: <Where your procedure is located>
   - *Stored Procedure*: <The stored procedure created that can process the input data>
   - *Include Original Input Data*: Default
   - *Result Handling*: Default
4. On the right-hand side of the screen, configure the tabs of the remote function adapter.
   - *Input Function*: <Enter the input function that the SAP HANA stored procedure needs to process>
     For the purposes of this example, it is the prerequisite model table “RFA Input” that was created earlier .
   - The *Rule* tab has two sections:
     1. *Template* Section (Read Only)
        This is a generated procedure based on the input function that can be used as a template for the called SAP HANA stored procedure.
        The configurer must be aware of the fact that this template is a mandatory format, and that it must be closely checked against the called SAP HANA stored procedure

        Example

        Sample Code

        ```
        CREATE PROCEDURE "<SCHEMA>"."<HANAPROCEDURE>"
        (IN it_al "<DEFAULT SCHEMA>"."/NXI/TP1AL",
        IN it_input TABLE(JB_PROD NVARCHAR(50)),
        OUT ot_result TABLE(JB_PROD NVARCHAR(50), FS_PER_MSG_TEXT_
        NVARCHAR(5000), FS_PER_FORMULA_ NVARCHAR(5000)),
        OUT ot_msg TABLE(MSGTY NVARCHAR(1), MSG_TEXT NVARCHAR(5000))
        ) LANGUAGE SQLSCRIPT SQL SECURITY DEFINER AS
        BEGIN
        END;
        ```
     2. *Statement* Section (Read Only)
        This reflects the SQL statement(s) of the called SAP HANA stored procedure.
        The parameter section of the SAP HANA stored procedure must be completely filled using the respective template format based on the input function.
        In this example, it looks something like this:

        Sample Code

        ```
        CREATE PROCEDURE "<SCHEMA>"."<HANAPROCEDURE>"
        (IN it_al "<DEFAULT SCHEMA>"."/NXI/TP1AL",
        IN it_input TABLE(JB_PROD NVARCHAR(50)),
        OUT ot_result TABLE(JB_PROD NVARCHAR(50), FS_PER_MSG_TEXT_
        NVARCHAR(5000), FS_PER_FORMULA_ NVARCHAR(5000)),
        OUT ot_msg TABLE(MSGTY NVARCHAR(1), MSG_TEXT NVARCHAR(5000))
        ) LANGUAGE SQLSCRIPT SQL SECURITY DEFINER AS
        ```

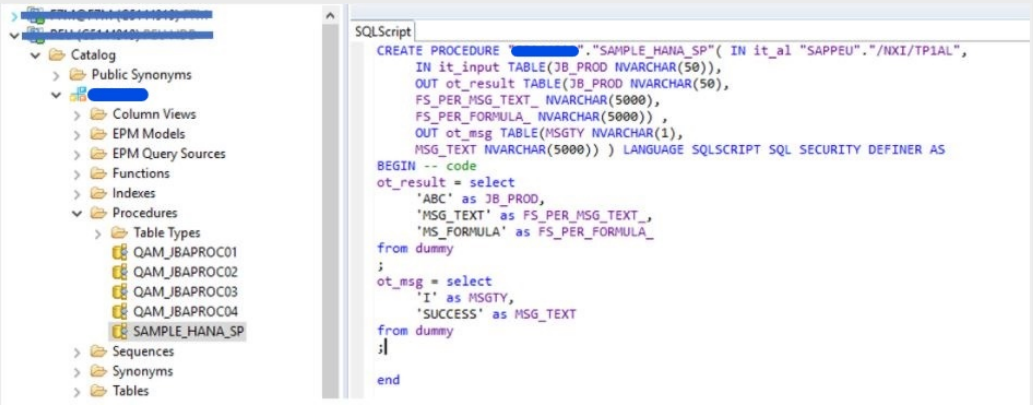
```
BEGIN
ot_result = select 'ABC' as JB_PROD, 'MSG_TEXT' as FS_PER_MSG_TEXT_,
'MS_FORMULA' as FS_PER_FORMULA_ from dummy;
ot_msg = select  'I'  as MSGTY, 'SUCCESS' as MSG_TEXT from dummy;
END;
```

Note

Explanation

- `ot_result` = This is the section where input processing will happen with the help of the SAP HANA stored procedure.
  In the above example, the procedure states that the value “ABC” is assigned to the field `JB_PROD`, assign “MSG_TEXT” to field `FS_PERMSG_TEXT_`, and add “MS_FORMULA” as a value for `FS_PER_FORMULA_` field.
  Since the last two fields are technical fields, the scenario can focus on `JB_PROD` now getting a value “ABC”.
- `ot_msg` = This is the section where run information can be manipulated and set by the SAP HANA stored procedure.
  In the above example, the procedure states that the value “I” or “Information” is assigned as `MSGTY` (msgtype), and have a word “SUCCESS” as `MSG_TEXT`.

Additional Note: This is an actual example of a SAP HANA procedure that SAP Profitability and Performance Management can call in the remote function adapter (this is the same as with the *Statement* section in the *Rule* tab above).

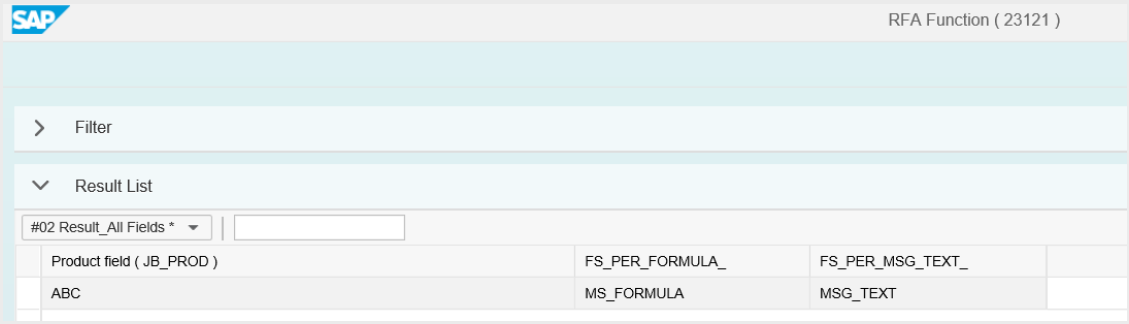


3. *Check*: Making entries on this tab is not mandatory, but if a check is required after processing, you can enter proper check conditions on this tab.
4. *Documentation*: Making entries on this tab is not mandatory, but you can use it to document the scenario or provide documentation or instructions about the function being modeled.

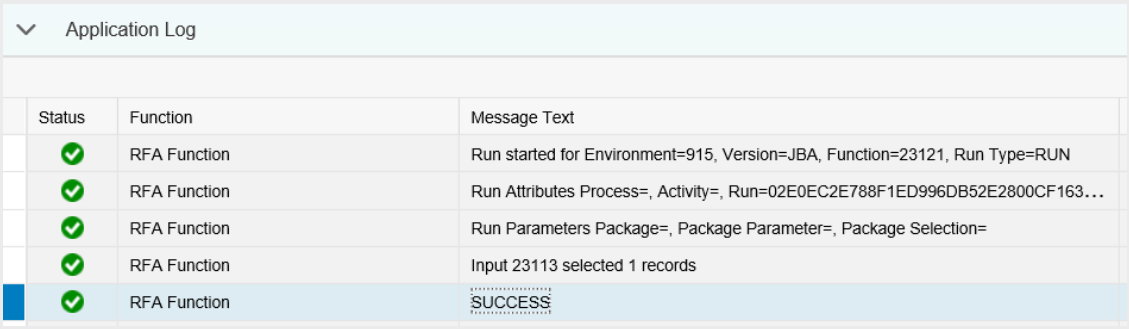
5. Once you have completed the set-up, choose *Activate*.
6. Then choose *Run*.

Example

Here is the result of the remote function adapter function after calling a SAP HANA stored procedure to process the input function:



Here is the application log of the remote function adapter function after calling a SAP HANA stored procedure to process the input function:



1.4.3.3 Writer

Writer is a processing function that allows you to write in the *Model Tables*, *Model BWs*, and *Model RDL* components. Therefore, the *Writer* function covers all the technical complexity of the different access modes.

Key Features

Header

You first need to define the output function. For this you can use all the model tables, model BWs and model RDL components of the environment.

> **i Note**
>
> The processing type should always be set to “Executable” in the function attributes.

The function automatically detects the type of output function and offers dependent choices.

1.4.3.3.1 Output: Model Table

When the output function is a Model Table, you can select one of the following Model Writer types:

- Insert [page 325].
- Delete and Insert [page 326].

> **i Note**
>
> Remember to do the following when you use this kind of output function type:
>
> - Select appropriate key fields in the Output Model table.
> - Use "Group by" conditions for the proper determination of key fields.
> - Use unique constraints, where applicable.

During activation, the HANA stored procedure is generated based on the following:

- Selected input and output functions
- Selections and formulas (if any) defined in the *Input & Output* tab of the *Writer* function.

During the run, the system behavior depends on the Model Writer type:

- When the Model Writer type is "Insert", the system executes the generated HANA stored procedure and inserts or aggregates the new data to already existing data in the output Model Table.

  > **i Note**
  >
  > The "Group by" statement defined in the output groups or aggregates the data with the same characteristic values. The aggregation behavior of the key figures can be defined using a formula (for example, `SUM (AMOUNT)`, `COUNT (PRODUCT)`).

- When the Model Writer type is "Delete and Insert", the system first deletes the data from the output Model Table and then inserts the new data. In the function attributes, the processing type should always be set to "Executable".

  > **i Note**
  >
  > When the writer is triggered from an iterative view, the system deletes data only once from the Model Table in the pre-run step, and not in each iteration.

Procedure

Function Access

Follow the steps below to access the *Writer* function with output "Model Table":

1. In the client where SAP Profitability and Performance Management is installed, choose *SAP Menu* > *Profitability and Performance Management* > *Modeling* > *Start My Environments* .
2. The *Environment* screen appears in a separate browser window. Choose an existing environment and continue. Within the environment, you can set up the newly added function.

Function Configuration

Follow the steps below to configure the *Writer* function with output “Model Table”:

1. In edit mode, configure the following required fields in the header. For more information about the header, see the *Key Features* section above.
   - *Output Function*
   - *Model Writer Type*: Choose either “Insert” or “Delete and Insert”
2. Define the input function to be used on the *Input* tab.

   > **i Note**
   >
   > For more information about the *Input* tab, see Input [page 11].
3. Define the fields to be used on the *Signature* tab.

   > **i Note**
   >
   > For more information about the *Signature* tab, see Signature [page 13].
4. Define the fields to be used on the *Output* tab.
   To do this, proceed as follows:
   1. Choose + (*Add*).
   2. The *Fields* screen is displayed. Select the fields to be added, then choose *OK*.

   > **i Note**
   >
   > For more information about the *Output* tab, see Output [page 15].
5. Define the checks to be used on the *Checks* tab, if needed.

   > **i Note**
   >
   > For more information about the *Checks* tab, see Checks [page 17].
6. Choose *Save* and then choose *Activate*.

Related Information

- For more information about common aspects of SAP Profitability and Performance Management functions, see Concepts for Key Users [page 6].
- For more information on how to add and remove functions in SAP Profitability and Performance Management environments, see Function Hierarchy [page 63].

1.4.3.3.1.1 Example: Model Writer Type "Insert"

All the data below is shown as it is displayed on the *Analyze* tab of the Model Table.

In the output function used for illustration here, the Model Table source is “Environment”.

Input Function

Customer	Product	Quantity	Amount
CN001	PROD01	50	500
CN002	PROD02	100	1000
CN002	PROD03	200	2000
CN003	PROD01	250	2500

Output Function

Before Run

Customer	Product	Quantity	Amount
CN001	PROD01	11	111
CN002	PROD02	22	222
CN003	PROD02	33	333
CN004	PROD05	55	555

After Run

Customer	Product	Quantity	Amount
CN001	**PROD01**	**61**	**611**
CN002	**PROD02**	**122**	**1222**
CN002	**PROD03**	**200**	**2000**
CN003	**PROD01**	**250**	**2500**
CN003	PROD02	33	333
CN004	PROD05	55	555

1.4.3.3.1.2 Example: Model Writer Type "Delete and Insert"

All the data below is shown as it is displayed on the *Analyze* tab of the Model Table.

In the output function used for illustration here, the Model Table source is “Environment”.

Input Function

Customer	Product	Quantity	Amount
CN001	PROD01	50	500
CN002	PROD02	100	1000
CN002	PROD03	200	2000
CN003	PROD01	250	2500

Output Function

Before Run

Customer	Product	Quantity	Amount
CN001	PROD01	11	111
CN002	PROD02	22	222
CN003	PROD02	33	333
CN004	PROD05	55	555

After Run – Without Process Selection

When the function is executed without process selection, the system first deletes all data and then adds new data.

Customer	Product	Quantity	Amount
CN001	**PROD01**	**50**	**500**
CN002	**PROD02**	**100**	**1000**
CN002	**PROD03**	**200**	**2000**
CN003	**PROD01**	**250**	**2500**

After Run – With Process Selection

When the same function is executed with process selection, for example "Product = PROD01, PROD03", the system first deletes only the data for PROD01 and PROD03 based on your selection and then adds the new data for PROD01 and PROD03. The following table shows the run results with the selection condition "Product = PROD01, PROD03".

Customer	Product	Quantity	Amount
CN001	**PROD01**	**50**	**500**
CN002	PROD02	22	222
CN002	**PROD03**	**200**	**2000**

Customer	Product	Quantity	Amount
CN003	**PROD01**	**250**	**2500**
CN003	PROD02	33	333

1.4.3.3.2 Output: Model BW

When the output function is a Model BW, you can select one of the following BW Writer types:

- Loading [page 328]
- Planning [page 331]

1.4.3.3.2.1 BW Writer Type "Loading"

Data is written using the data transfer process (DTP), and users who are editing data cannot continue to work during this time.

The model writer type “Insert” is available. For more information, see Example: Model Writer Type "Insert" [page 330].

During activation, the following BW objects are generated (for more information, see *Runtime Attributes*):

- Data source
- Transformation
- Data transfer process
- Based on the type of Model BW, required process types are added to the process chain.

Real-Time Load Behavior

1. If the Model BW source is “Environment”, after the run the real-time load behavior is set either to:
   - Loading mode if the *Editable* option is “No”
   - Planning mode if the *Editable* option is “Yes”.
2. If the Model BW source is “Business Warehouse” and BW Infoprovider is “Planning ADSO”, the real-time load behavior is always set to “Planning” mode after the run.

Activation of Data

1. If the Model BW source is “Business Warehouse” (BW ADSOs and Classic DSOs), which requires data activation, data is activated automatically in the process chain when loading is finished, and is available for reporting immediately.
2. If the Model BW source is “Business Warehouse” (ADSOs , such as “Infocube”, created), data is immediately available for reporting once loading is finished. No data activation takes place here.

During the run, the generated process chain is executed, which loads the data from the input function to the output function using the data transfer process (DTP).

Procedure

Function Access

Follow the steps below to access the *Writer* function with output "Model BW" and BW writer type "Loading":

1. In the client where SAP Profitability and Performance Management is installed, choose *SAP Menu* > *Profitability and Performance Management* > *Modeling* > *Start My Environments*.
2. The *Environment* screen appears in a separate browser window. Choose an existing environment and continue. Within the environment, you can set up the newly added function.

Function Configuration

Follow the steps below to configure the *Writer* function with output "Model BW" and BW writer type "Loading":

1. In edit mode, configure the following required fields in the header. For more information about the header, see the *Key Features* section above.
   - *Output Function* = "Model BW"
   - *BW Writer Type* = "Loading"
   - *Model Writer Type* = "Insert"
2. Define the input function to be used on the *Input* tab.

   > **i Note**
   >
   > For more information about the *Input* tab, see Input [page 11].

3. Define the fields to be used on the *Signature* tab.

   > **i Note**
   >
   > For more information about the *Signature* tab, see Signature [page 13].

4. Define the fields to be used on the *Output* tab, if needed.
   To do this, proceed as follows:
   1. Choose + (*Add*).
   2. The *Fields* screen is displayed. Select the fields to be added, then choose *OK*.

   > **i Note**
   >
   > For more information about the *Output* tab, see Output [page 15].

5. Define the checks to be used on the *Checks* tab, if needed.

   > **i Note**
   >
   > For more information about the *Checks* tab, see Checks [page 17].

6. Choose *Save* and then choose *Activate*.

Related Information

- For more information about common aspects of SAP Profitability and Performance Management functions, see Concepts for Key Users [page 6].
- For more information on how to add and remove functions in SAP Profitability and Performance Management environments, see Function Hierarchy [page 63].

1.4.3.3.2.1.1 Example: Model Writer Type "Insert"

In the output function used for illustration here, the Model BW source is “Environment”.

All the data below is shown as it is displayed on the *Analyze* tab of the Model BW.

New records are inserted and the existing records are overwritten in the Output Model BW.

Execution of Loading Writer Function

Input Function

Customer	Product	Quantity	Amount
CN001	PROD01	50	500
CN002	PROD02	100	1000
CN002	PROD03	200	2000
CN003	PROD01	250	2500

Output Function – Before Run

Customer	Product	Quantity	Amount
CN001	PROD01	11	111
CN002	PROD02	22	222
CN003	PROD02	33	333
CN004	PROD05	55	555

Output Function – After Run

Customer	Product	Quantity	Amount
CN001	**PROD01**	**50**	**500**
CN002	**PROD02**	**100**	**1000**
CN002	**PROD03**	**200**	**2000**
CN003	**PROD01**	**250**	**2500**

Customer	Product	Quantity	Amount
CN003	PROD02	33	333
CN004	PROD05	55	555

The first two rows have been overwritten, whereas the third and fourth row have been inserted.

Execution of Loading Writer Function with Process ID Filter Selection

Extraction mode in the DTP is always set to “Full” from the datasource to Model BW. When the *Writer* function is executed with a different process selection, the DTP filter condition is applied to the current process ID and the data is filtered and loaded to the Model BW.

> **i** Note
>
> If DTP is executed manually or from a different process chain and not called by the writer function, all data is extracted from the data source and can result in data duplication.

1.4.3.3.2.2 BW Writer Type "Planning"

Data is written to output function objects using the Planning Engine and users who are editing data can continue to work.

The following model writer types are availalable:

- Insert [page 333]
- Modify [page 334]
- Delete and Insert [page 335]

During activation, the following BW objects are generated (for more information, see *Runtime Attributes*):

- Aggregation level
- Filter
- Planning function
- Planning sequence

Additional Information

- If the installations of BW and SAP Profitability and Performance Management are in different clients, you need to define the callback RFC destination to execute the planning sequence in the BW client (T-code: `RSPLAN`).
  To define the *Callback RFC* destination, go to the *Advanced* tab of the *My Environments* screen. For more information, see the Advanced [page 63] section.

- The planning sequence or planning function created in SAP Profitability and Performance Management can be used directly in any of the BW reporting tools, such as Analysis for Office (AAO), Lumira or SAP Analytics for Cloud (SAC).

Procedure

Function Access

Follow the steps below to access the *Writer* function with output "Model BW" and BW writer type "Planning":

1. In the client where SAP Profitability and Performance Management is installed, choose *SAP Menu* > *Profitability and Performance Management* > *Modeling* > *Start My Environments*.
2. The *Environment* screen appears in a separate browser window. Choose an existing environment and continue. Within the environment, you can set up the newly added function.

Function Configuration

Follow the steps below to configure the *Writer* function with output "Model BW" and BW writer type "Planning":

1. In edit mode, configure the following required fields in the header. For more information about the header, see the *Key Features* section above.
   - *Output Function* = "Model BW"
   - *BW Writer Type* = "Planning"
   - *Model Writer Type*: Choose either "Insert", "Modify" or "Delete and Insert".
2. Define the input function to be used on the *Input* tab.

   > **i Note**
   >
   > For more information about the *Input* tab, see Input [page 11].

3. Define the fields to be used on the *Signature* tab.

   > **i Note**
   >
   > For more information about the *Signature* tab, see Signature [page 13].

4. Define the fields to be used on the *Output* tab, if needed.
   To do this, proceed as follows:
   1. Choose + (*Add*).
   2. The *Fields* screen is displayed. Select the fields to be added, then choose *OK*.

   > **i Note**
   >
   > For more information about the *Output* tab, see Output [page 15].

5. Define the checks to be used on the *Checks* tab, if needed.

   > **i Note**
   >
   > For more information about the *Checks* tab, see Checks [page 17].

6. Choose *Save* and then choose *Activate*.

Related Information

- For more information about common aspects of SAP Profitability and Performance Management functions, see Concepts for Key Users [page 6].
- For more information on how to add and remove functions in SAP Profitability and Performance Management environments, see Function Hierarchy [page 63].

1.4.3.3.2.2.1 Example: Model Writer Type "Insert"

Data is inserted or aggregated with already existing data in the Output Model BW.

All the data below is shown as it is displayed on the *Analyze* tab of the Model BW.

In the output function used for illustration here, the Model BW source is “Environment”.

Input Function

Customer	Product	Quantity	Amount
CN001	PROD01	50	500
CN002	PROD02	100	1000
CN002	PROD03	200	2000
CN003	PROD01	250	2500

Output Function

Before Run

Customer	Product	Quantity	Amount
CN001	PROD01	11	111
CN002	PROD02	22	222
CN003	PROD02	33	333
CN004	PROD05	55	555

After Run

Customer	Product	Quantity	Amount
CN001	**PROD01**	**61**	**611**
CN002	**PROD02**	**122**	**1222**

Customer	Product	Quantity	Amount
CN002	**PROD03**	**200**	**2000**
CN003	**PROD01**	**250**	**2500**
CN003	PROD02	33	333
CN004	PROD05	55	555

1.4.3.3.2.2.2 Example: Model Writer Type "Modify"

Data with the same characteristic values is overwritten and new data is inserted.

All the data below is shown as it is displayed on the *Analyze* tab of the Model BW.

In the output function used for illustration here, the Model BW source is “Environment”.

Input Function

Customer	Product	Quantity	Amount
CN001	PROD01	50	500
CN002	PROD02	100	1000
CN002	PROD03	200	2000
CN003	PROD01	250	2500

Output Function

Before Run

Customer	Product	Quantity	Amount
CN001	PROD01	11	111
CN002	PROD02	22	222
CN003	PROD02	33	333
CN004	PROD05	55	555

After Run

Customer	Product	Quantity	Amount
CN001	**PROD01**	**50**	**500**

Customer	Product	Quantity	Amount
CN002	**PROD02**	**100**	**1000**
CN002	**PROD03**	**200**	**2000**
CN003	**PROD01**	**250**	**2500**
CN003	PROD02	33	333
CN004	PROD05	55	555

1.4.3.3.2.2.3 Example: Model Writer Type "Delete and Insert"

The existing data is not physically deleted from the Model BW. Instead, all key figure values of the existing data are either converted to the value “0” or zeroed out by inserting records with negative values for the key figures. New data is then appended or aggregated to the existing data.

All the data below is shown as it is displayed on the *Analyze* tab of the Model BW.

In the output function used for illustration here, the Model BW source is “Environment”.

Input Function

Customer	Product	Quantity	Amount
CN001	PROD01	50	500
CN002	PROD02	100	1000
CN002	PROD03	200	2000
CN003	PROD01	250	2500

Output Function

Before Run

Customer	Product	Quantity	Amount
CN001	PROD01	11	111
CN002	PROD02	22	222
CN003	PROD02	33	333
CN004	PROD05	55	555

After Run – Without Process Selection

Customer	Product	Quantity	Amount
CN001	**PROD01**	**50**	**500**
CN002	**PROD02**	**100**	**1000**
CN002	**PROD03**	**200**	**2000**
CN003	**PROD01**	**250**	**2500**

After Run – With Process Selection

When the same function is executed with process selection (for example, "Product EQ PROD01, PROD03"), the system first deletes only the data for PROD01 and PROD03 based on your selection, and then adds new data for PROD01 and PROD03. The following table shows the run results with the selection condition "Product = PROD01, PROD03".

Customer	Product	Quantity	Amount
CN001	**PROD01**	**50**	**500**
CN002	PROD02	22	222
CN002	**PROD03**	**200**	**2000**
CN003	**PROD01**	**250**	**2500**
CN003	PROD02	33	333

> **i** Note
>
> For better system performance, we recommend that you always use input, output or process selections wherever possible. These selections are propagated to the planning engine, which reduces initial data load when the planning function is executed. If no selection is added, the planning engine selects all data from the Infoprovider, which may result in OOM (out of memory) errors.

1.4.3.3.3 Output: Model RDL

SAP Profitability and Performance Management writes data records to a specific RDL result type using a *Writer* function with a configured output function pointing to a Model RDL. This *Model RDL* function must then be configured to point to a specific results data area and result type.

The system writes to these result types by calling AMDP classes that have active SAP HANA procedures. There are specific result categories where AMDP classes automatically generate SAP HANA procedures. Therefore, as a prerequisite for writing to specific result types, these need to be assigned to one of the following result categories: HKRIC, HSRIC, HKCFR, HKEPS, HSEPS, HKACG, HSACG, HKAMS, HSAMS, HKANA, HSANA, HKRMC, HSRMC, HKVOL, HSVOL, HKMES, HSMES, HKPAT, HSPAT, HKLSE, HSLSE, HKLFP, HSLFP, HKPAP, HSPAP, HKPAI, HSPAI, HKRPE, HSRPE, HKRPS, HSRPS, HKULT, HSULT, HKVEC, HSVEC, HKCFO, HKCDA.

If the connection is established to FPSL RDLs, this is reflected in the Customizing activity *Financial Products Subledger* > *Data Model* > *Edit Result Data Area and Data Structures*. Choose *Result Data Area Result Types* to see if the result type is mapped to the accepted result category (*ResCat*).

For more information, see SAP Note 2935308 – PaPM 3.0 Bank Analyzer/Insurance Analyzer Result Data Layer integration.

1.4.4 Structuring Functions

SAP Profitability and Performance Management offers the following functions which help you to structure environments or models:

- Calculation Unit [page 337]
  The *Calculation Unit* function helps to structure larger environments into multiple parts that can define processes and activities independently of each other. A typical example is to use it to structure a decentralized month-end closing process into separate business units and/or closing units (= calculation units), where each closing unit has its own processes and activities. Only the results are stored in one central place at the end.
- Description [page 346]
  The *Description* function helps to structure and document the model. It does not affect the result but improves the readability of the model.

1.4.4.1 Calculation Unit

A calculation unit represents a financial or business unit on which calculations and analysis like financial closing can be performed independently of other calculation units.

A calculation unit is a container function that defines the functions relevant for execution using process templates and activities. It also specifies the parameters and selection fields that are required for execution.

From a modeling perspective, it is a collection of objects, such as fields and functions.

Key Features

Process Templates and Activities

Process templates are used to specify the set of functions that are relevant for execution. A process template is structured by one or many activities which have to be executed to finish the process.

Process templates and activities are defined by the following information:

1. *Process Template ID*
   Calculation unit-wide unique ID of a process template that can be referred to in process management to instantiate processes.
2. *Description*
   Short description of the purpose of a process template.
3. *Process Template State*

The process template state can be set to either inactive or active. Inactive process templates are not ready to be deployed. Active process templates can be deployed and instantiated as a process in process management, and can therefore be used to run processes in production.

4. *Process Type*
   Can be set to either “Simulation” or “Run”. If the process type is set to “Simulation”, all parameters and field selections can be changed at any time to allow what-if simulation. If the process type is set to “Run”, all parameters and field selections have to be fixed during the deployment and cannot be changed during what-if simulation.
5. *Activity ID*
   Process template-wide unique ID of an activity that can be referred to in report elements.
6. *Description*
   Short description of the purpose of an activity.
7. *Activity Type*
   The activity type can be set to either “Input/Output” or “Execution”. “Input/Output” allows you to look at the data of a function, typically a query function. “Execution” allows you to trigger the execution of a function.
8. *Level*
   The level defines which activities depend on each other.
   You can set up activities in hierarchical form by using the *Same Level* or *One Level Below* options when you add them on the *Activities* tab.

   > **Example**
   >
   > For example, the activity *Review Actual Data* with level 1 can be worked on immediately after process deployment, but the activity *Execution Calculation* underneath it with level 2 will remain pending until the activity *Review Actual Data* is finished.

9. *Checks*
   When the *Activity Type* is “Input/Output”, you can assign one or more checks either of category “Including Check” or “Excluding Check” to a process activity. The checks maintained here are used to validate the activity result data during the run in the *My Activities* application. They also influence the Dual Control activities *Submit* and *Complete*.
   For more information about the impact of checks in *My Activities*, see Dual Control [page 343].
10. *Start Date* and *End Date*
    In both fields, you need to define default values. These can be overwritten during process deployment.
11. *Performer* and *Reviewer*
    The performer defines a team (group of users) that can work on an activity. The reviewer can also define a team that has to review the activity in a workflow using the Dual Control [page 343] principle and can either approve or reject it.

Parameters

Parameters are defined in the environment and can be registered here so that they are available for use in process templates. They can influence the behavior of functions below the calculation unit at runtime.

> **Example**
>
> You want to analyze the profitability of an organizational unit every quarter using an assumed sector growth rate %. For this, the modeling user can design a business model on the basis of a *Period* parameter. The financial analyst (execution user) can specify this parameter value during the deployment of a process.

Selection Fields

Fields are defined in the environment and can be registered here as selection fields so that they are available for use in process templates. Selection fields can filter the input data of functions below the calculation unit at runtime.

> **Example**
>
> You want to analyze the profitability of an organizational unit every quarter. For this, the modeling user can design a business model on the basis of a *Period* selection. The financial analyst (execution user) can specify this selection value during the deployment of a process.

Business Event Fields

Fields are defined in the environment and can be registered here as business event fields so that the event and error handling for all functions in the calculation unit is done at that common level. If no business event fields are registered, error and event handling is done for the individual fields of each function.

Documentation

User-specific inline documentation can be entered here to describe which settings were made and why.

Procedures

In the client where SAP Profitability and Performance Management is installed, choose *SAP Menu* > *Profitability and Performance Management* > *Modeling* > *Start My Environments*.

The *Environment* screen appears in a separate browser window. Here you can create or choose an existing environment and continue. Once you are in the environment, choose the calculation unit and switch to *Edit* mode, which allows you to perform the following activities:

Process Template

Adding a Process Template

1. Choose *Add (+)*,
2. The *Add Details* window appears, and you can enter the attributes of the process template that you want to create.
3. Choose *OK* and the created process template is displayed.

Removing a Process Template

1. Select an existing process template that you want to delete.
   Choose *Remove (–)*.
   Choose *Save* after the deletion to finalize the changes you have made.

Copying a Process Template

You can copy an existing process template and thereby change the process template ID, description and target position. The system also copies all activities and selections present in the source template .

1. Select the existing process template you want to copy.
2. Choose *Copy*.

3. The *Copy Details* window appears.
4. Enter the new process template ID, description and target position.
5. Choose *OK*.
6. Choose *Save* once you have copied your entries to finalize the changes you have made.

Activities Inside Process Templates

Once you have selected the process template in edit mode, you can perform the following activities on the *Activities* tab:

Adding an Activity

1. Choose *Add (+)*,
2. The *Add Details* window appears.
3. When you have made your entries for the fields, choose *OK*.
4. Choose *Save* to complete the changes you have made.

Removing an Activity

1. Choose an existing activity that you want to delete.
2. Choose *Remove (–)*.
3. Choose *Save* after you have deleted to save the changes you have made.

Copying an Activity

You can also copy an existing activity. Parameters and Selections from the copied activity are then transferred to the new activity.

1. Select an existing activity that you want to copy.
2. Choose *Copy*
3. The *Copy Details* window appears.
4. Once you have entered the new activity ID, description and target position, choose *OK*.
5. Choose *Save* to save the changes you have made.

Selections Inside Process Template

Within the process template, you can process the following activities on the *Selections* tab in edit mode:

- Parameters
  - **Add**
    1. Choose *Add (+)*.
    2. The *Fields* window appears.
    3. The system lists the available parameters. Select the desired parameter and choose *OK*.
    4. Choose *Save* to complete the changes made.
  - **Remove**
    1. Choose a parameter that you want to delete.
    2. Choose *Remove (–)*
    3. Choose *Save* to save the changes you have made.
  - **Formula**
    Use this function to set the desired parameter value.
    1. Select a parameter and choose *(F)*.
    2. The *Formula* window appears.

3. Enter the required value for the parameter and choose *OK*.
4. Choose *Save* to save the changes you have made.

- Selection Conditions
  - **Add**
    1. Choose *Add (+)*.
    2. The *Fields* window appears.
    3. Select the required selection field and choose *OK*.
    4. Choose *Save* to save the changes you have made.
  - **Remove**
    1. Select a field that you want to delete.
    2. Choose *Remove (–)*.
    3. Choose *Save* to save the changes you have made.
  - **Filter**
    1. Select a field to configure the selection.
    2. Choose *Filter*.
    3. The *Selection Condition* window appears.
    4. Enter the required selection value for the field using operators.
    5. Choose *OK*.
    6. Choose *Save* to save the changes you have made.
  - **Paste Filter Selection**
    You can use this function to copy a filter selection from the reporting application.
    1. Select an existing field.
    2. Choose *Paste Filter Selection*.
    3. The system copies the selection filter to the selected field.

Parameters

Adding a Parameter

1. Choose *Add (+)*,
2. The system dispays the parameter fields available in the environment.

> **i** Note
>
> The parameters declared here are listed in the *Activity Selection* section.

3. Select the required parameter field and choose *OK*.

Removing a Parameter

1. Select the parameter that you want to delete.
2. Choose *Remove (–)*.
3. Choose *Save* to save the changes you have made.

Selection

Adding a Selection

1. Choose *Add (+)*,
2. The system displays the fields available in the environment.

> **Note**
>
> Fields declared here are listed in the *Activity Selection Condition* section.

3. Select the required field and choose *OK*.

Removing a Selection

1. Select the field you want to delete.
2. Choose *Remove (–)*.
3. Choose *Save* to save the changes you have made.

Business Event Fields

Adding a Field

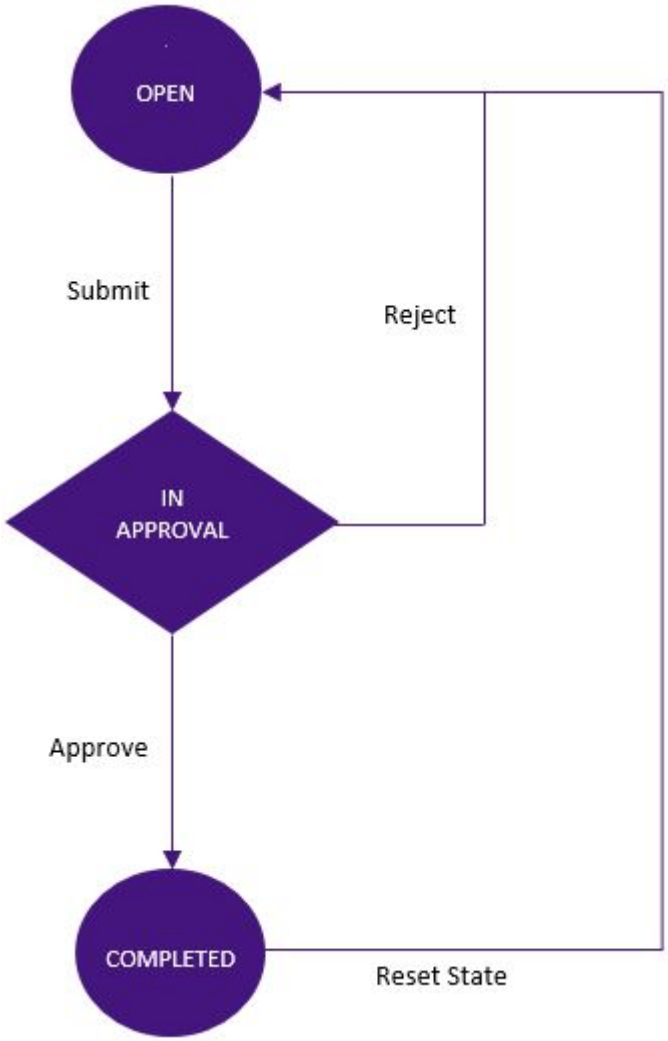
1. Choose *Add (+)*.
2. The *Add Fields* window appears. It displays the fields available in the environment.
3. Select the required field and choose *OK*.

Removing a Field

1. Select the field that you want to delete.
2. Choose *Remove (–)*.
3. Choose *Save* to save the changes you have made.

1.4.4.1.1 Dual Control

Dual control enables two user groups (performer and reviewer) to execute the actions *Submit*, *Approve*, *Reject* and *Reset State* for activities according to the Dual Control Workflow shown below.



i Note

- Once an activity is rejected, the system resets the status to “Open”, and you need to change it before you can submit the activity again.
- If a user belongs to the performer group and reviewer group, the user can only act either as performer or reviewer, but never both.

  #### Example

  If you submit an activity for approval, you are not allowed to act as the reviewer to approve or reject this activity, even though you are actually part of the reviewer group assigned to the activity.

- If both the *Performer* and the *Reviewer* field of an activity is empty, the dual control mechanism is not active. Only the *Complete* button is enabled.

- If the *Reviewer* field of an activity is filled and the *Performer* field is empty, the dual control mechanism is not active. In this case, the *Complete* button is only enabled for users who are part of the performer group.
- Checks are performed on *Complete* and *Submit* actions if they are maintained for process activities (input/output only) and if the check condition is met. Otherwise, the following scenarios apply, depending on the *Check Message Type*:
  - For the message types "Error" and "Abort", the system stops the actions *Complete* or *Submit*. It provides an error message to inform you that the check has failed. In this case, it is not possible to complete or submit the activity.
  - For other message types ("Information", "Status" and "Warning"), the system allows you to perform the actions *Complete* or *Submit* even if the check condition is not met.
- In all of the cases above, the system provides a message text for the corresponding check. The message text is maintained on environment level in the system message overview.

If an activity has a defined previous activity, the system sets the parent activity state to "Pending" until the previous activities are completed. Afterwards, the system follows the flowchart. The example below gives a step-by-step overview of how the status of the activities changes:

Initial State

Activity	Status	Previous Activity
A0001	Pending	A0003
A0002	Pending	A0003
A0003	Pending	A0004
A0004	Open	

When the performer submits activity A0004 for review, the status changes from "Open" to "In Approval":

Activity	Status	Previous Activity
A0001	Pending	A0003
A0002	Pending	A0003
A0003	Pending	A0004
A0004	**In Approval**	

As soon as activity A0004 has been approved, the status changes from "In Approval" to "Completed". The status of activity A0003 is now "Open":

Activity	Status	Previous Activity
A0001	Pending	A0003
A0002	Pending	A0003
A0003	**Open**	A0004
A0004	Completed	

When the performer submits activity A0003 for review, the status changes from “Open” to “In Approval”:

Activity	Status	Previous Activity
A0001	Pending	A0003
A0002	Pending	A0003
A0003	**In Approval**	A0004
A0004	Completed	

As soon as Activity A0003 has been approved, the status changes from “In Approval” to “Completed”. The status of the activities A0001 and A0002 is now “Open”:

Activity	Status	Previous Activity
A0001	Open	A0003
A0002	Open	A0003
A0003	**Completed**	A0004
A0004	Completed	

When the performer submits activity A0001 for review, the status changes from “Open” to “In Approval”:

Activity	Status	Previous Activity
A0001	**In Approval**	A0003
A0002	Open	A0003
A0003	Completed	A0004
A0004	Completed	

When the performer submits activity A0002 for review, the status changes from “Open” to “In Approval”:

Activity	Status	Previous Activity
A0001	In Approval	A0003
A0002	**In Approval**	A0003
A0003	Completed	A0004
A0004	Completed	

As soon as activity A0001 has been approved, the status changes from “In Approval” to “Completed”:

Activity	Status	Previous Activity
A0001	**Completed**	A0003
A0002	In Approval	A0003
A0003	Completed	A0004
A0004	Completed	

As soon as activity A0002 has been approved, the status changes from “In Approval” to “Completed”:

Activity	Status	Previous Activity
A0001	Completed	A0003
A0002	**Completed**	A0003
A0003	Completed	A0004
A0004	Completed	

> **i** Note
>
> SAP Profitability and Performance Management sends email notifications in the following cases:
>
> - Activity sent for approval, receiver is reviewer team
> - Activity approved, receiver is both performer and reviewer team
> - Activity rejected, receiver is both performer and reviewer team.

1.4.4.2 Description

A function that provides information, usually used to explain or provide details about a function or hierarchy of functions. It is used for the inline documentation of models as well as to structure other functions in the modeling hierarchy.

> **i** Note
>
> The *Description* function is also a part of each function. You can find it on the *Documentation* tab.

Documentation Editor

You can use the editor and its simple formatting options to provide descriptions and documentation about the function configuration and why things where modeled this way.

You can also use the following buttons to formatthe texts in the documentation editor:

Button	Use
Bold	Darkens the text to emphasize it.
Italic	Formats the text in a slightly slanted way to set it apart.
Increase Indent	Increases space between the current paragraph and the left page margin.
Decrease Indent	Decreases space between the current paragraph and the left page margin.
Numbering	Creates an ordered list for steps or checklists.

Button	Use
Bullets	Creates an unordered list to denote significance.
Heading 1, Heading 2 and Heading 3	Formats text as a (section) title.

Related Information

For more details about common aspects of SAP Profitability and Performance Management functions, see Concepts for Key Users [page 6].

1.4.5 Query Function

The *Query* function defines whether the data display for a user is read-only or editable. For the editable query, the input function has to be a *Model BW* function. A query function does not provide output to subsequent functions because its purpose is data input (when editable) and reporting (when read-only and editable).

- Query [page 347]

1.4.5.1 Query

Query is a function that is used for defining the report and input-enabled queries (planning). It can be used to represent the input function data in the form of tables and charts in the *Analyze* section.

The following query sources are managed entirely in SAP Profitability and Performance Management:

- Environment [page 350]
- Environment CDS [page 357]

The following query sources and workbooks are managed externally:

- Analysis for Office [page 348]
- Business Warehouse [page 349]

These can also be accessed in SAP Profitability and Performance Management.

Key Features

Header

In the header, you first need to define the query source. The function provides options depending on the selected query source.

Related Information

For more information about common aspects of SAP Profitability and Performance Management functions, see Concepts for Key Users [page 6].

1.4.5.1.1 Query Source: Analysis for Office

If you select “Analysis for Office” as your query source, workbooks that are managed in SAP Business Warehouse can be accessed in SAP Profitability and Performance Management to make them available in the environment.

Header

If you have selected “Analysis for Office” as the query source, the function automatically allows you to select from the list of workbooks already defined in SAP BW.

Procedure

Function Access

Follow the steps below to access the *Query* function with query source “Analysis for Office”:

1. In the client where SAP Profitability and Performance Management is installed, choose *SAP Menu* > *Profitability and Performance Management* > *Modeling* > *Start My Environments*.
2. The *Environment* screen appears in a separate browser window. Choose an existing environment and continue. Within the environment, you can set up the newly added function.

Function Configuration

Follow the steps below to configure the *Query* function with query source “Analysis for Office”:

1. In edit mode, configure the following required fields in the header.
   - *Query Source* = “Analysis for Office”
   - *Workbook*: Choose a file as an input.

   > **i Note**
   >
   > Depending on the chosen query source the header shows different fields.
2. Choose *Save* and then choose *Activate*.

Related Information

- For more information about common aspects of SAP Profitability and Performance Management functions, see Concepts for Key Users [page 6].
- For more information on how to add and remove functions in SAP Profitability and Performance Management environments, see Function Hierarchy [page 63].

1.4.5.1.2 Query Source: Business Warehouse

If you select “Business Warehouse” as the query source, queries that are managed in SAP Business Warehouse can be accessed in SAP Profitability and Performance Management to make them available in the environment.

Header

If you have selected “Business Warehouse” as the query source, the function automatically allows you to select from the list of queries already defined in SAP BW.

> **i** Note
>
> The *Show Advanced Search* field allows you to search the query based on its technical name, the InfoProvider, author and version.

Procedure

Function Access

Follow the steps below to access the *Query* function with query source “Business Warehouse”:

1. In the client where SAP Profitability and Performance Management is installed, choose *SAP Menu* > *Profitability and Performance Management* > *Modeling* > *Start My Environments*.
2. The *Environment* screen appears in a separate browser window. Choose an existing environment and continue. Within the environment, you can set up the newly added function.

Function Configuration

Follow the steps below to configure the *Query* function with query source “Business Warehouse”:

1. In edit mode, configure the following required fields in the header.
   - *Query Source* = “Business Warehouse”
   - *Query*: Choose the file to be used as input.

   > **i** Note
   >
   > Depending on the chosen query source the header shows different fields.

2. Choose *Save* and then choose *Activate*.

Related Information

- For more information about common aspects of SAP Profitability and Performance Management functions, see Concepts for Key Users [page 6].
- For more information on how to add and remove functions in SAP Profitability and Performance Management environments, see Function Hierarchy [page 63].

1.4.5.1.3 Query Source: Environment

When you select "Environment" as the query source, the definition of the *Query* function is managed entirely in the SAP Profitability and Performance Management environment. Technically, this uses NetWeaver and BEx Query technology. During activation, the system creates a BEx query that can be used in SAP Profitability and Performance Management or externally in the SAP BW reporting tools.

Input Function

You select the input function based on the query which is created, and underlying data is used for reporting or planning purposes.

> **i** Note
>
> An executable function or *Model* function can only be used as input for a query.

Key Date (Optional)

The master data (attributes, texts and hierarchies) of time-dependent InfoObjects is displayed in the output based on the key date that you specify.

> **i** Note
>
> The key date only applies to time-dependent master data.

Editable

By default, this option is disabled and set to "No".

When the selected input function is *Model BW* and allows planning, the *Editable* option is enabled.

- If you select “Yes”, the system allows both reporting and planning. Manual data input or planning is permitted.
  If you select “No”, the system allows only reporting. Manual data input or planning is not permitted.

Filter Tab

This tab defines general fixed and default values. You can specify a variable to enter values manually when running the query.

- **Fixed Values**: The query output is filtered and displayed based on the values defined in the selection or the values entered for the variables. You cannot change these values in the output results.
- **Default Values**: The query output is initially displayed based on the values defined in the selection or the values entered for the variables. In the results that the system returns, you can change these values in the filter option.

> **i** Note
>
> Variables are always input-ready and prompt you to enter values when the query is executed (with the exception of authorization variables, which are processed automatically).

The sections below are displayed dynamically, based on the characteristic you have selected:

- *General*: Displays the technical name and description.
- *Selection*: An option to specify predefined filter values.
- *Variable*: An option to enter the filter values in the *Variable* screen when the query is executed. Variables can be defined in the following ways:
  - “Single Value”: Only one value is allowed (can be mandatory or optional).
  - “Selection Option”: Single value, multiple single values, a range or a combination of these are allowed (can be mandatory or optional).
  - Authorization: This is available only for SAP BW InfoObjects which are authorization-relevant.

> **i** Note
>
> When you execute the report from *My Activities*, the system fills the variable valuesin the *Analyze* screen with the appropriate selections defined in the process. You can execute a query with these default values or by changing the values.

Sheet Definition

In the sheet definition, you define the rows, columns and free fields.

> **i** Note
>
> Free fields are shown under *Available Fields* in the output and can be used for further analysis.

The following definitions are available:

- **Characteristic**: Can be added to a column or row definition.
- **Key figure**: Can be added to a column or row definition.
- **Selection**: You can define "restricted" columns or rows by choosing particular characteristic values. You can also choose a key figure and restrict it to one or more characteristic values.
- **Formula**: Allows you to define any calculations that are executed during a report run. For example, the summation of two or more fields or complex calculations using formula functions. Only the functions available in the SAP BEx Query Designer are supported here.

  > **i Note**
  >
  > When you define the formula, the operands and operators should be separated by spaces.
  >
  > Example: KF01 / (KF02 – KF03)

> **i Note**
>
> Key figures, formulas and selections can only be included under a structure.

For each **Characteristic** the following settings are available:

- **General**
  Displays the general properties of a field or characteristic when running a query:
  - *Description*: Displays the description of the field in the report.

    > **i Note**
    >
    > The description can only be changed in the *Add Details* dialog when a characteristic is added to the query definition.

  - *Query Component*: Represents the technical name of the field or characteristic.
- **Hierarchy**
  This is available only when the selected characteristic has a hierarchy defined in the SAP Profitability and Performance Management environment.
  - *Hierarchy Name*: If the field has active hierarchies, you can choose one as a default for the report. You can choose another hierarchy during runtime.
  - *Hierarchy Date*: If the hierarchies are time-dependent, you must specify a constant here.
  - *Hierarchy Version*: If the hierarchies are versioned, you must specify a constant here.
- **Value Output Format**
  This section is available only for Environment/BW Infoobjects. It allows you to define how the master data of the characteristic is displayed when running a query.
  - Display As:
    - "No Display": The characteristic display is hidden. This function is particularly useful with the currency or unit characteristic for example, since the currencies are also displayed in the key figures.
    - "Key and Text": The characteristic values are displayed with their technical key and their text.
    - "Text": The characteristic values are displayed with their text.
    - "Key": The characteristic values are displayed with their technical key.
    - "Text and Key": The characteristic values are displayed with their text and their technical key.

If you set a display type that contains text, the following options are available (provided you have made the corresponding settings in the InfoObject master data):

- Text Output
  - "Standard": The shortest available text for the characteristic values is used.
  - "Short Text": The short text for the characteristic values is used.
  - "Long Text": The long text for the characteristic values is used.
  - "Medium Text": The medium text for the characteristic values is used.

- **Result Output Format**
  Show Result Rows allows you to define whether result rows for particular characteristics are displayed when running a query:
  - "Always": Result rows are always displayed.
  - "Never": Result rows are always suppressed.
  - Only if more than one child exists, are result rows shown when characteristics have more than one row displayed in the report output.
- **Extended**
  This section is available only for Environment or BW InfoObjects.
  Access Types for Result Values allows you to define the values shown in the value help:
  - "Posted Values": The system only displays values that are available in the underlying InfoProvider.
  - "Master Data": The system only displays values that are available in the characteristic master data.
  - "Characteristic Relationships": The system only displays these if they have been defined in the SAP BW Planning Modeler.

For each **Key Figure** the following settings are available:

- **General**
  Defines the general properties of key figures when running a query:
  - *Description*: You can modify the description of the key figure to be displayed in the report.
  - *Query Component*: Represents the technical name of the key figure.
- **Key Figure**
  Aggregation behavior:
  - "Default": The aggregation behavior is taken from the environment field or key figure definition.
  - "Maximum": In case of aggregation, the key figure maximum value is displayed.
  - "Minimum": In case of aggregation, the key figure minimum value is displayed.
  - "Summation": In case of aggregation, the key figures are summed for display.
- **Editable**
  This is available only when *Editable* is set to "Yes" in the header. You can select individual key figures for planning or the manual input of data. You can select one of the following options:
  - "Yes": Manual data input or planning is permitted.
  - "No": Manual data input or planning is not permitted.
- **Display**
  - *Hide*: Defines the key figure's display behavior
    - Always Show
    - Always Hide
    - Hide (can be shown): You can hide or unhide these key figures in the report output.
  - *Highlight*
    - Normal display

    - Highlighted display: Can be seen only in external reporting tools (for example, Analysis for Office)
- *Number of Decimal Places*: Defines the number of decimal places to be displayed for individual key figures in the report output.
- *Scaling Factor*: Defines the scaling factor of the individual key figures displayed.
- *Change Sign*
    - "Keep +/– Sign": Displays the key figure value as stored.
    - "Reverse Sign": Displays the reverse sign of the stored key figure value.

Analyze Screen

On the *Analyze* screen of an editable query, you can use the *Data Grid Upload* feature. This feature allows you to upload data from a Microsoft Excel file directly on the *Analyze* query level.

> **i** Note
>
> - The Data Grid Upload always works in overwrite mode, which means that it zeroes all existing values and upload values from the Excel file.
> - Filters (both on fixed values and default values) are not applicable with this feature. Hence, it always uploads all existing records from the Excel file.

Required structure of the file:

- The first row of data must contain mapping information (technical names of fields that are used in the input function).
- Actual data that you want to upload must start from the second row.
- Duplicated records will be summed up.
- Decimal places of key figures must be separated by: "."
- Thousands should be separated with ",", however this is not mandatory
- Every date has to be set in the following format: DD.MM.YYYY
- The following file formats are supported: *.xlsx, *.xlsm

Procedure

Function Access

Follow the steps below to access the *Query* function with query source "Environment":

1. In the client where SAP Profitability and Performance Management is installed, choose *SAP Menu* > *Profitability and Performance Management* > *Modeling* > *Start My Environments* .
2. The *Environment* screen appears in a separate browser window. Choose an existing environment and continue. Within the environment, you can set up the newly added function.

Function Configuration

Follow the steps below to configure the *Query* function with query source “Environment”:

1. In edit mode, configure the following required fields in the header.
   - *Query Source* = “Environment”
   - *Input Function*
   - *Editable*
   - *Key Date*

   > **i** Note
   >
   > Depending on the chosen query source the header shows different fields.

2. Define the fields on the *Fixed Values* and *Default* sections on the *Filter* tab as follows:
   1. Choose + (*Add*).
   2. The *Add Details* screen appears. Assign the following parameters:
      - *Rule*: Descriptive rule name
      - *Query Component*: Technical name of the field
      - *Description*: Name of the field
   3. Choose *OK*.
   4. Fill out the required fields that appear on the right-hand side of the query function when you add the fields.
3. Define the *Columns*, *Rows* and *Free* section on the *Sheet Definition* tab.
   To do this proceed as follows:
   1. Choose + (*Add*).
   2. The *Add Details* screen appears. Assign the following parameters:
      - *Rule*: Descriptive rule name
      - *Rule Type*: Technical name of the field
      - *Description*: Name of the field
   3. Choose *OK*.
   4. Fill out the required fields that appear on the right-hand side of the query function when you add the fields.
4. Choose *Save* and then choose *Activate*.

Related Information

- For more information about common aspects of SAP Profitability and Performance Management functions, see Concepts for Key Users [page 6].
- For more information on how to add and remove functions in SAP Profitability and Performance Management environments, see Function Hierarchy [page 63].

1.4.5.1.3.1 Example: Query Source "Environment"

This example shows how you can create a simple query and define that the key figures are editable.

Input Function

Customer	Product	Quantity	Amount
CN001	PROD01	50	500
CN002	PROD02	100	1000
CN002	PROD03	200	2000
CN003	PROD01	250	2500

Fixed Values

You can add the field *Customer* and select “Selection option” and “Optional” in the *Variable* section.

Default Values

You can add the field *Product* and select the value “PROD01” in the *Selection* section.

Columns:

You first define the key figure structure and then include the key figure formula. You can select individual key figures and set the *Editable* indicator to “Yes”.

Name	Rule Type	Editable
KYF	STR	
Quantity	KYF	Yes
Amount	KYF	Yes
Price per Unit	FML = Amount/Quantity	

Rows:

Name	Rule Type
Customer	CHA
Product	CHA

Free:

This is left empty.

Query Output

Customer	Product	Quantity	Amount	Price per Unit
CN001	PROD01	50	500	10
CN003	PROD01	250	2500	10

You can change the values of the key figures *Quantity* and *Amount* and save these values.

> **i Note**
>
> The system filters the result to display only the productPROD01. Clear the filter to display all of the data available.

1.4.5.1.4 Query Source: Environment CDS

If you select “Environment CDS” as the query source, the definition of the *Query* function is managed entirely in the SAP Profitability and Performance Management environment. Technically, this uses SAP ABAP CDS technology. During activation, the system creates an analytical query CDSview that can be used in SAP Profitability and Performance Management or externally in the SAP BW reporting tools.

Input Function

You select the input function based on the query which is created. Underlying data is used for reporting or planning purposes.

> **i Note**
>
> An executable function or *Model* function with a CDS view as the InfoProvider can only be used as input for a query.

Editable

By default, this option is disabled and set to “No”.

If you select the input function *Model* *Table Source* *Environment*, the *Editable* option is enabled.

- If you select “Yes” , the system allows both reporting and planning. Manual data input or planning is permitted.
  If you select “No” , the system allows only reporting. Manual data input or planning is not permitted.

Filter Tab

This tab defines general fixed and default values. You can specify a variable to enter values manually when you run the query.

For fixed values, the query output is filtered and displayed based on the values defined in the selection or the values entered for the variables. You cannot change these values in the output results.

Variables are always input-ready and prompt you to enter values when the query is executed.

The sections below are displayed dynamically, based on the characteristics you select:

- *General*: Displays the technical name and description.
- *Selection*: An option to specify predefined filter values.
- *Variable*: An option to enter the filter values on the *Variable* screen during query execution. Variables can be defined in the following ways:
  - "Single Value": Only one value is allowed (can be mandatory or optional).
  - "Selection Option": Single value, multiple single values, a range or a combination of these are allowed (can be mandatory or optional).
- *Variable Default Value*: You can enter the default value for the variable.

> **i** Note
>
> When you execute the report in *My Activities*, the system fills the variable values on the *Analyze* screen with the appropriate selections defined in the process. You can execute a query with these default values or by changing the values.

Sheet Definition

In the sheet definition you define the rows, columns and free fields.

> **i** Note
>
> Free fields are shown under *Available Fields* in the output and can be used for further analysis.

The following definitions are available:

- **Characteristic**: Can be added to a column or row definition.
- **Key figure**: Can be added to column or row definition.
- **Selection**: Allows you to define "restricted" columns or rows by choosing particular characteristic values. You can also choose a key figure and restrict it to one or more characteristic values.
- **Formula**: Allows you to define any calculations that are executed during the report run (or example, the summation of two fields or complex expressions).

  > **i** Note
  >
  > When you define the formula, the operands and operators should be separated by spaces.
  >
  > Example: KF01 / (KF02 – KF03)

The following settings are available for each characteristic :

- **General**
  Displays the general properties of a field or characteristic when running a query:
  - *Description*: Displays the description of the field in the report.

    > **i** Note
    >
    > The description can only be changed in the *Add Details* dialog when a characteristic is added to the query definition.

- *Query Component*: Represents the technical name of the field or characteristic.

- **Hierarchy**
  This is available only if you have defined a hierarchy for the selected characteristic in the SAP Profitability and Performance Management environment.
  - *Hierarchy Name*: If the field has active hierarchies, you can choose one as a default for the report. You can choose another hierarchy during runtime.
- **Value Output Format**
  This section is available only for Environment/BW InfoObjects. It allows you to define how the master data of the characteristic is displayed when you run a query.
  - Display As:
    - "Key and Text": The characteristic values are displayed with their technical key and their text.
    - "Text": The characteristic values are displayed with their text.
    - "Key": The characteristic values are displayed with their technical key.

  If you set a display type that contains text, the following options are available (provided you have made the appropriate settings in the InfoObject master data):
  - Text Output
    - "Standard": The shortest available text for the characteristic values is used.
    - "Long Text": The long text for the characteristic values is used.
- **Result Output Format**
  Show Result Rows allows you to define whether result rows for particular characteristic are displayed when your run a query:
  - "Always": Result rows are always displayed.
  - "Never": Result rows are always hidden.
  - Only if more than one child node exists, are result rows displayed when characteristics have more than one row displayed in the report output.

The following settings are available for each key figure:

- **General**
  Defines general properties of key figures when running a query:
  - *Description*: You can modify the description of the key figure to be displayed in the report.
  - *Query Component*: Represents the technical name of the key figure.
- **Key Figure**
  Aggregation behavior:
  - "Default": Summation is considered by default.
  - "Maximum": In case of aggregation, the key figure maximum value is displayed.
  - "Minimum": In case of aggregation, the key figure minimum value is displayed.
  - "Summation": In case of aggregation, the key figure is summed for display.
- Editable
  This is available only if you have set the *Editable* indicator to "Yes" in the header. Individual key figures can be selected for planning or the manual input of data. You can select one of the following options:
  - "Yes": Manual data input or planning is permitted.
  - "No": Manual data input or planning is not permitted.
- **Display**
  - *Hide*: Defines the display behavior of the key figure.
    - Always Show

- Always Hide
- *Number of Decimal Places*: Defines the number of decimal places to be displayed for individual key figures in the report output.
- *Scaling Factor*: Defines the scaling factor of the individual key figures displayed.

Analyze Screen

On the *Analyze* screen of an editable query, you can use the *Data Grid Upload* feature. This feature allows you to upload data from a Microsoft Excel file directly on the *Analyze* query level.

> **i Note**
>
> - The Data Grid Upload always works in overwrite mode, which means that it zeroes all existing values and upload values from the Excel file.
> - Filters (both on fixed values and default values) are not applicable with this feature. Hence, it always uploads all existing records from the Excel file.

Required structure of the file:

- The first row of data must contain mapping information (technical names of fields that are used in the input function).
- Actual data that you want to upload must start from the second row.
- Duplicated records will be summed up.
- Decimal places of key figures must be separated by: "."
- Thousands should be separated with ",", however this is not mandatory
- Every date has to be set in the following format: DD.MM.YYYY
- The following file formats are supported: *.xlsx, *.xlsm

Procedure

Function Access

Follow the steps below to access the *Query* function with query source "Environment CDS":

1. In the client where SAP Profitability and Performance Management is installed, choose *SAP Menu* > *Profitability and Performance Management* > *Modeling* > *Start My Environments*.
2. The *Environment* screen appears in a separate browser window. Choose an existing environment and continue. Within the environment, you can set up the newly added function.

Function Configuration

Follow the steps below to configure the *Query* function with query source "Environment CDS":

1. In edit mode, configure the following required fields in the header.
   - *Query Source* = "Environment CDS"
   - *Input Function*

- *Editable*

> **i Note**
>
> Depending on the chosen query source the header shows different fields.

2. Define the fields on the *Fixed Values* section on the *Filter* tab as follows:
   1. Choose + (*Add*).
   2. The *Add Details* screen appears. Assign the following parameters:
      - *Rule*: Descriptive rule name
      - *Query Component*: Technical name of the field
      - *Description*: Name of the field
   3. Choose *OK*.
   4. Fill out the required fields that appear on the right-hand side of the query function when you add the fields.
3. Define the *Columns*, *Rows* and *Free* section on the *Sheet Definition* tab.
   To do this proceed as follows:
   1. Choose + (*Add*).
   2. The *Add Details* screen appears. Assign the following parameters:
      - *Rule*: Descriptive rule name
      - *Rule Type*: Technical name of the field
      - *Description*: Name of the field
   3. Choose *OK*.
   4. Fill out the required fields that appear on the right-hand side of the query function when you add the fields.
4. Choose *Save* and then choose *Activate*.

Related Information

- For more information about common aspects of SAP Profitability and Performance Management functions, see Concepts for Key Users [page 6].
- For more information on how to add and remove functions in SAP Profitability and Performance Management environments, see Function Hierarchy [page 63].

1.4.6 Analytics Function

The *Machine Learning* function is the systematic computational analysis of data or statistics. It is used for the discovery, interpretation and communication of meaningful patterns in data. Furthermore, it entails applying data patterns towards effective decision making.

- Machine Learning [page 362]

1.4.6.1 Machine Learning

Machine learning is an application of artificial intelligence (AI) that provides systems with the ability to automatically learn and improve from experience without being explicitly programmed.

The *Machine Learning* function allows you to get knowledge and insights and learn from your own data within SAP Profitability and Performance Management. By completing a few simple configuration steps, you can train and deploy best-in-class predictive models that currently deal with classification, clustering, regression and time-series forecasting scenarios. You can leverage these functionalities powered by SAP automated predictive technology to integrate trustworthy predictions into your business process models built in or upon SAP Profitability and Performance Management, so that you augment their business intelligence capabilities by learning from data.

Key Features

Header

All header options such as *Result Handling*, *Include Original Input Data*, *Suppress Initial Results* and *Result Model Table* are called "Functional Building Blocks" and are not specific to Machine Learning. For more information about these options, see Header [page 10].

Rules

Machine Learning offers the following rule types:

- Forecast [page 364]
- Clustering [page 377]
- Regression [page 378]
- Classification [page 379]
- Recommendation [page 380]

Procedure

Function Access

Follow the steps below to access the *Machine Learning* function:

1. In the client where SAP Profitability and Performance Management is installed, choose *SAP Menu* > *Profitability and Performance Management* > *Modeling* > *Start My Environments* .
2. The *Environment* screen appears in a separate browser window. Choose an existing environment and continue. Within the environment, you can set up the newly added function.

Function Configuration

Follow the steps below to configure the *Machine Learning* function:

1. In edit mode, configure the following required fields in the header. For more information about the header, see the *Key Features* section above.

- *Result Handling*
- *Include Original Input Data*
- *Suppress Initial Result*
- *Result Model Table*

2. Define the input function to be used on the *Input* tab.

> **i Note**
>
> For more information about the *Input* tab, see Input [page 11].

3. Define the fields to be used on the *Signature* tab.

> **i Note**
>
> For more information about the *Signature* tab, see Signature [page 13].

4. Define the rules on the *Rules* tab as follows:
    1. Choose + (*Add*).
    2. The *Add Details* screen appears. Configure the following parameters:
        - *Add Level*
        - *Rule*
        - *Rule Type*
        - *Description*
    3. Choose *OK*.
    4. The *Rule Lines* section appears comprising the *Input Fields* and *Output Fields* sections. Fill out the mandatory fields marked by an asterisk (*).
    5. You can also add fields on the *Subview* tab, if needed. To do this, proceed as follows:
        1. Choose + (*Add*).
        2. The *Fields* screen is displayed. Select a field and choose *OK*.
        3. Assign a formula or a condition for the field.

> **i Note**
>
> Depending on the chosen rule type, the rule lines show different configurations.

5. Define the checks to be used on the *Checks* tab, if needed.

> **i Note**
>
> For more information about the *Checks* tab, see Checks [page 17].

6. Choose *Save* and then choose *Activate*.

Related Information

- For more information about common aspects of SAP Profitability and Performance Management functions, see Concepts for Key Users [page 6].
- For more information on how to add and remove functions in SAP Profitability and Performance Management environments, see Function Hierarchy [page 63].

1.4.6.1.1 Rule Type: Forecast

The *Machine Learning* function provides the rule type *Forecast* to train and use time-series predictive models based on input data. It runs several models (for example, linear regression, or exponential smoothing) on historical data to determine the best model trained from the input dataset. It also predicts future values with this model for a specific measure. The forecasted values can be used later in other functions. The prediction is only applied to the specific selected measure. However, input data is replaced in the output if you choose to run a forecast over a field that contains historical data.

Rule Input Fields

You can use the following input fields for configuration:

- *Date Field*: Specifies the date field for the time series
- *End Date*: Specifies the end date as a constant or parameter of the historical data
- *Signal Field*: Specifies the field from which the values are used for the forecast
- *Excluded Fields*: List of fields that are not relevant for the forecast, or those fields whose impact should be excluded from the forecast
- *Forecast Period*: Period for which the forecast is executed
- *Forecast Unit*: Period unit for the number of periods (for example, year, quarter, or month)
- *Forecast Method*: Specifies the forecasting algorithm

  > **i** Note
  >
  > The *Forecast Method* field, which is originally set to “Default”, may be explicitly changed to “Linear Regression” or “Exponential Smoothing” to provide a more realistic result as it may happen at times that the algorithm generates unexpected results or duplicated forecasted values.

- *Positive Forecast*: Defines whether only positive forecasted values are generated
- *Segmented By*: Defines which fields are segmented based on the input dataset in case of multiple independent forecasts

Rule Output Fields

You can use the following output fields for configuration:

- *Forecast Field*: The field in which the system writes the forecast result from the signal field
- *Model*: An optional field containing the trained or used model ID that is automatically assigned by the forecast algorithm. It is a characteristic field type and we recommend you use a length of 30 characters.

1.4.6.1.1.1 Example: Rule Type "Forecast"

Input Data

Date	Valid To	Approved Amount	Y	Historical Data
2014-01-31	2014-01-31	7,915,648.000	1	A
2014-02-28	2014-02-28	8,117,755.340	1	A
2014-03-31	2014-03-31	8,137,351.860	1	A
2014-04-30	2014-04-30	8,613,248.970	1	A
2014-05-31	2014-05-31	9,125,681.200	1	A
2014-06-30	2014-06-30	8,937,232.630	1	A
2014-07-31	2014-07-31	9,509,737.920	1	A
2014-08-31	2014-08-31	9,850,823.910	1	A
2014-09-30	2014-09-30	9,670,498.720	1	A
2014-10-31	2014-10-31	9,723,401.650	1	A
2014-11-30	2014-11-30	9,891,196.600	1	A
2014-12-31	2014-12-31	9,606,154.580	1	A
2015-01-31	2015-01-31	9,771,109.530	1	A
2015-02-28	2015-02-28	10,894,153.580	1	A
2015-03-31	2015-03-31	10,806,004.680	1	A
2015-04-30	2015-04-30	10,900,457.090	1	A
2015-05-31	2015-05-31	11,114,289.780	1	A
2015-06-30	2015-06-30	11,634,529.420	1	A
2015-07-31	2015-07-31	11,746,286.450	1	A
2015-08-31	2015-08-31	11,619,104.800	1	A
2015-09-30	2015-09-30	12,526,260.650	1	A
2015-10-31	2015-10-31	12,539,040.390	1	A
2015-11-30	2015-11-30	12,605,884.700	1	A
2015-12-31	2015-12-31	12,084,553.320	1	A
2016-01-31	2016-01-31	12,857,099.610	1	A
2016-02-29	2016-02-29	13,691,459.800	1	A
2016-03-31	2016-03-31	13,616,689.930	1	A
2016-04-30	2016-04-30	13,868,179.430	1	A
2016-05-31	2016-05-31	13,868,179.430	1	A

Date	Valid To	Approved Amount	Y	Historical Data
2016-06-30	2016-06-30	13,922,138.610	1	A
2016-07-31	2016-07-31	14,331,946.890	1	A
2016-08-31	2016-08-31	14,897,174.960	1	A
2016-09-30	2016-09-30	14,807,930.210	1	A
2016-10-31	2016-10-31	14,786,580.090	1	A
2016-11-30	2016-11-30	14,840,931.360	1	A
2016-12-31	2016-12-31	15,772,728.090	1	A
2017-01-31	2017-01-31	15,104,910.860	1	A
2017-02-28	2017-02-28	15,196,866.190	1	A
2017-03-31	2017-03-31	16,717,905.220	1	A
2017-04-30	2017-04-30	16,320,447.320	1	A
2017-05-31	2017-05-31	16,273,357.880	1	A
2017-06-30	2017-06-30	16,683,750.080	1	A
2017-07-31	2017-07-31	16,335,991.190	1	A
2017-08-31	2017-08-31	17,438,468.070	1	A
2017-09-30	2017-09-30	17,031,429.660	1	A
2017-10-31	2017-10-31	17,914,564.640	1	A
2017-11-30	2017-11-30	16,713,307.780	1	A
2017-12-31	2017-12-31	17,684,150.310	1	A
2018-01-31	2018-01-31	18,088,397.820	1	A
2018-02-28	2018-02-28	17,647,209.960	1	A
2018-03-31	2018-03-31	17,177,858.280	1	A
2018-04-30	2018-04-30	19,117,232.130	1	A
2018-05-31	2018-05-31	19,455,425.620	1	A
2018-06-30	2018-06-30	19,442,605.340	1	A
2018-07-31	2018-07-31	19,500,040.890	1	A
2018-08-31	2018-08-31	18,567,658.220	1	A
2018-09-30	2018-09-30	18,982,459.030	1	A
2018-10-31	2018-10-31	20,718,186.800	1	A
2018-11-30	2018-11-30	19,841,863.110	1	A
2018-12-31	2018-12-31	21,120,797.180	1	A
2019-01-31	2019-01-31	21,477,047.410	1	A
2019-02-28	2019-02-28	19,307,530.800	1	A
2019-03-31	2019-03-31	21,056,419.990	1	A

Date	Valid To	Approved Amount	Y	Historical Data
2019-04-30	2019-04-30	21,564,104.630	1	A
2019-05-31	2019-05-31	21,130,564.960	1	A
2019-06-30	2019-06-30	20,934,401.670	1	A
2019-07-31	2019-07-31	23,245,440.940	1	A
2019-08-31	2019-08-31	22,066,748.240	1	A
2019-09-30	2019-09-30	22,148,297.710	1	A
2019-10-31	2019-10-31	21,527,433.260	1	A
2019-11-30	2019-11-30	22,925,550.220	1	A
2019-12-31	2019-12-31	22,722,256.830	1	A
2020-01-31	2020-01-31	18,849,520.440	1	A
2020-02-29	2020-02-29	18,786,380.170	1	A
2020-03-31	2020-03-31	18,434,899.660	1	A
2020-04-30	2020-04-30	18,564,151.680	1	A
2020-05-31	2020-05-31	18,501,521.020	1	A
2020-06-30	2020-06-30	19,708,359.490	1	A
2020-07-31	2020-07-31	18,829,774.560	1	A
2020-08-31	2020-08-31	18,814,294.240	1	A
2020-09-30	2020-09-30	18,924,925.620	1	A
2020-10-31	2020-10-31	19,971,153.280	1	A
2020-11-30	2020-11-30	19,721,535.780	1	A
2020-12-31	2020-12-31	18,800,017.170	1	A
2021-01-31	2021-01-31	19,446,307.750	1	A
2021-02-28	2021-02-28	19,973,412.180	1	A
2021-03-31	2021-03-31	19,043,403.310	1	A
2021-04-30	2021-04-30	21,122,491.670	1	A
2021-05-31	2021-05-31	20,478,794.610	1	A
2021-06-30	2021-06-30	20,340,826.070	1	A

The historical data run from 31-01-2014 to 30-06-21, so the end date is specified as “20210630”. The forecast period is set to 12 as the function generates the forecasted values in the *Approved Amount* column for the next 12 months based on the algorithm of the Automated Predictive Learning being applied.

Result – Forecast Method: Default

Date	Valid To	Approved Amount	Y	Historical Data
2014-01-31	2014-01-31	7,915,648.000	0	A
2014-02-28	2014-02-28	8,117,755.340	0	A
2014-03-31	2014-03-31	8,137,351.860	0	A
2014-04-30	2014-04-30	8,613,248.970	0	A
2014-05-31	2014-05-31	9,125,681.200	0	A
2014-06-30	2014-06-30	8,937,232.630	0	A
2014-07-31	2014-07-31	9,509,737.920	0	A
2014-08-31	2014-08-31	9,850,823.910	0	A
2014-09-30	2014-09-30	9,670,498.720	0	A
2014-10-31	2014-10-31	9,723,401.650	0	A
2014-11-30	2014-11-30	9,891,196.600	0	A
2014-12-31	2014-12-31	9,606,154.580	0	A
2015-01-31	2015-01-31	9,771,109.530	0	A
2015-02-28	2015-02-28	10,894,153.580	0	A
2015-03-31	2015-03-31	10,806,004.680	0	A
2015-04-30	2015-04-30	10,900,457.090	0	A
2015-05-31	2015-05-31	11,114,289.780	0	A
2015-06-30	2015-06-30	11,634,529.420	0	A
2015-07-31	2015-07-31	11,746,286.450	0	A
2015-08-31	2015-08-31	11,619,104.800	0	A
2015-09-30	2015-09-30	12,526,260.650	0	A
2015-10-31	2015-10-31	12,539,040.390	0	A
2015-11-30	2015-11-30	12,605,884.700	0	A
2015-12-31	2015-12-31	12,084,553.320	0	A
2016-01-31	2016-01-31	12,857,099.610	0	A
2016-02-29	2016-02-29	13,691,459.800	0	A
2016-03-31	2016-03-31	13,616,689.930	0	A
2016-04-30	2016-04-30	13,868,179.430	0	A
2016-05-31	2016-05-31	13,868,179.430	0	A
2016-06-30	2016-06-30	13,922,138.610	0	A
2016-07-31	2016-07-31	14,331,946.890	0	A
2016-08-31	2016-08-31	14,897,174.960	0	A
2016-09-30	2016-09-30	14,807,930.210	0	A

Date	Valid To	Approved Amount	Y	Historical Data
2016-10-31	2016-10-31	14,786,580.090	0	A
2016-11-30	2016-11-30	14,840,931.360	0	A
2016-12-31	2016-12-31	15,772,728.090	0	A
2017-01-31	2017-01-31	15,104,910.860	0	A
2017-02-28	2017-02-28	15,196,866.190	0	A
2017-03-31	2017-03-31	16,717,905.220	0	A
2017-04-30	2017-04-30	16,320,447.320	0	A
2017-05-31	2017-05-31	16,273,357.880	0	A
2017-06-30	2017-06-30	16,683,750.080	0	A
2017-07-31	2017-07-31	16,335,991.190	0	A
2017-08-31	2017-08-31	17,438,468.070	0	A
2017-09-30	2017-09-30	17,031,429.660	0	A
2017-10-31	2017-10-31	17,914,564.640	0	A
2017-11-30	2017-11-30	16,713,307.780	0	A
2017-12-31	2017-12-31	17,684,150.310	0	A
2018-01-31	2018-01-31	18,088,397.820	0	A
2018-02-28	2018-02-28	17,647,209.960	0	A
2018-03-31	2018-03-31	17,177,858.280	0	A
2018-04-30	2018-04-30	19,117,232.130	0	A
2018-05-31	2018-05-31	19,455,425.620	0	A
2018-06-30	2018-06-30	19,442,605.340	0	A
2018-07-31	2018-07-31	19,500,040.890	0	A
2018-08-31	2018-08-31	18,567,658.220	0	A
2018-09-30	2018-09-30	18,982,459.030	0	A
2018-10-31	2018-10-31	20,718,186.800	0	A
2018-11-30	2018-11-30	19,841,863.110	0	A
2018-12-31	2018-12-31	21,120,797.180	0	A
2019-01-31	2019-01-31	21,477,047.410	0	A
2019-02-28	2019-02-28	19,307,530.800	0	A
2019-03-31	2019-03-31	21,056,419.990	0	A
2019-04-30	2019-04-30	21,564,104.630	0	A
2019-05-31	2019-05-31	21,130,564.960	0	A
2019-06-30	2019-06-30	20,934,401.670	0	A
2019-07-31	2019-07-31	23,245,440.940	0	A

Date	Valid To	Approved Amount	Y	Historical Data
2019-08-31	2019-08-31	22,066,748.240	0	A
2019-09-30	2019-09-30	22,148,297.710	0	A
2019-10-31	2019-10-31	21,527,433.260	0	A
2019-11-30	2019-11-30	22,925,550.220	0	A
2019-12-31	2019-12-31	22,722,256.830	0	A
2020-01-31	2020-01-31	18,849,520.440	0	A
2020-02-29	2020-02-29	18,786,380.170	0	A
2020-03-31	2020-03-31	18,434,899.660	0	A
2020-04-30	2020-04-30	18,564,151.680	0	A
2020-05-31	2020-05-31	18,501,521.020	0	A
2020-06-30	2020-06-30	19,708,359.490	0	A
2020-07-31	2020-07-31	18,829,774.560	0	A
2020-08-31	2020-08-31	18,814,294.240	0	A
2020-09-30	2020-09-30	18,924,925.620	0	A
2020-10-31	2020-10-31	19,971,153.280	0	A
2020-11-30	2020-11-30	19,721,535.780	0	A
2020-12-31	2020-12-31	18,800,017.170	0	A
2021-01-31	2021-01-31	19,446,307.750	0	A
2021-02-28	2021-02-28	19,973,412.180	0	A
2021-03-31	2021-03-31	19,043,403.310	0	A
2021-04-30	2021-04-30	21,122,491.670	0	A
2021-05-31	2021-05-31	20,478,794.610	0	A
2021-06-30	2021-06-30	20,340,826.070	0	A
2021-07-30		20,340,826.070	0	A
2021-08-30		20,340,826.070	0	A
2021-09-30		20,340,826.070	0	A
2021-10-30		20,340,826.070	0	A
2021-11-30		20,340,826.070	0	A
2021-12-30		20,340,826.070	0	A
2022-01-30		20,340,826.070	0	A
2022-02-28		20,340,826.070	0	A
2022-03-30		20,340,826.070	0	A
2022-04-30		20,340,826.070	0	A
2022-05-30		20,340,826.070	0	A

Date	Valid To	Approved Amount	Y	Historical Data
2022-06-30		20,340,826.070	0	A

In case of unexpected results that include duplicate forecasted values as shown above, you can explicitly change the algorithm through the input field *Forecast Method* in the rule line. In addition, there are the following options: “Linear Regression” and “Exponential Smoothing”.

Result – Forecast Method: Linear Regression

Date	Valid To	Approved Amount	Y	Historical Data
2014-01-31	2014-01-31	7,915,648.000	0	A
2014-02-28	2014-02-28	8,117,755.340	0	A
2014-03-31	2014-03-31	8,137,351.860	0	A
2014-04-30	2014-04-30	8,613,248.970	0	A
2014-05-31	2014-05-31	9,125,681.200	0	A
2014-06-30	2014-06-30	8,937,232.630	0	A
2014-07-31	2014-07-31	9,509,737.920	0	A
2014-08-31	2014-08-31	9,850,823.910	0	A
2014-09-30	2014-09-30	9,670,498.720	0	A
2014-10-31	2014-10-31	9,723,401.650	0	A
2014-11-30	2014-11-30	9,891,196.600	0	A
2014-12-31	2014-12-31	9,606,154.580	0	A
2015-01-31	2015-01-31	9,771,109.530	0	A
2015-02-28	2015-02-28	10,894,153.580	0	A
2015-03-31	2015-03-31	10,806,004.680	0	A
2015-04-30	2015-04-30	10,900,457.090	0	A
2015-05-31	2015-05-31	11,114,289.780	0	A
2015-06-30	2015-06-30	11,634,529.420	0	A
2015-07-31	2015-07-31	11,746,286.450	0	A
2015-08-31	2015-08-31	11,619,104.800	0	A
2015-09-30	2015-09-30	12,526,260.650	0	A
2015-10-31	2015-10-31	12,539,040.390	0	A
2015-11-30	2015-11-30	12,605,884.700	0	A
2015-12-31	2015-12-31	12,084,553.320	0	A
2016-01-31	2016-01-31	12,857,099.610	0	A
2016-02-29	2016-02-29	13,691,459.800	0	A

Date	Valid To	Approved Amount	Y	Historical Data
2016-03-31	2016-03-31	13,616,689.930	0	A
2016-04-30	2016-04-30	13,868,179.430	0	A
2016-05-31	2016-05-31	13,868,179.430	0	A
2016-06-30	2016-06-30	13,922,138.610	0	A
2016-07-31	2016-07-31	14,331,946.890	0	A
2016-08-31	2016-08-31	14,897,174.960	0	A
2016-09-30	2016-09-30	14,807,930.210	0	A
2016-10-31	2016-10-31	14,786,580.090	0	A
2016-11-30	2016-11-30	14,840,931.360	0	A
2016-12-31	2016-12-31	15,772,728.090	0	A
2017-01-31	2017-01-31	15,104,910.860	0	A
2017-02-28	2017-02-28	15,196,866.190	0	A
2017-03-31	2017-03-31	16,717,905.220	0	A
2017-04-30	2017-04-30	16,320,447.320	0	A
2017-05-31	2017-05-31	16,273,357.880	0	A
2017-06-30	2017-06-30	16,683,750.080	0	A
2017-07-31	2017-07-31	16,335,991.190	0	A
2017-08-31	2017-08-31	17,438,468.070	0	A
2017-09-30	2017-09-30	17,031,429.660	0	A
2017-10-31	2017-10-31	17,914,564.640	0	A
2017-11-30	2017-11-30	16,713,307.780	0	A
2017-12-31	2017-12-31	17,684,150.310	0	A
2018-01-31	2018-01-31	18,088,397.820	0	A
2018-02-28	2018-02-28	17,647,209.960	0	A
2018-03-31	2018-03-31	17,177,858.280	0	A
2018-04-30	2018-04-30	19,117,232.130	0	A
2018-05-31	2018-05-31	19,455,425.620	0	A
2018-06-30	2018-06-30	19,442,605.340	0	A
2018-07-31	2018-07-31	19,500,040.890	0	A
2018-08-31	2018-08-31	18,567,658.220	0	A
2018-09-30	2018-09-30	18,982,459.030	0	A
2018-10-31	2018-10-31	20,718,186.800	0	A
2018-11-30	2018-11-30	19,841,863.110	0	A
2018-12-31	2018-12-31	21,120,797.180	0	A

Date	Valid To	Approved Amount	Y	Historical Data
2019-01-31	2019-01-31	21,477,047.410	0	A
2019-02-28	2019-02-28	19,307,530.800	0	A
2019-03-31	2019-03-31	21,056,419.990	0	A
2019-04-30	2019-04-30	21,564,104.630	0	A
2019-05-31	2019-05-31	21,130,564.960	0	A
2019-06-30	2019-06-30	20,934,401.670	0	A
2019-07-31	2019-07-31	23,245,440.940	0	A
2019-08-31	2019-08-31	22,066,748.240	0	A
2019-09-30	2019-09-30	22,148,297.710	0	A
2019-10-31	2019-10-31	21,527,433.260	0	A
2019-11-30	2019-11-30	22,925,550.220	0	A
2019-12-31	2019-12-31	22,722,256.830	0	A
2020-01-31	2020-01-31	18,849,520.440	0	A
2020-02-29	2020-02-29	18,786,380.170	0	A
2020-03-31	2020-03-31	18,434,899.660	0	A
2020-04-30	2020-04-30	18,564,151.680	0	A
2020-05-31	2020-05-31	18,501,521.020	0	A
2020-06-30	2020-06-30	19,708,359.490	0	A
2020-07-31	2020-07-31	18,829,774.560	0	A
2020-08-31	2020-08-31	18,814,294.240	0	A
2020-09-30	2020-09-30	18,924,925.620	0	A
2020-10-31	2020-10-31	19,971,153.280	0	A
2020-11-30	2020-11-30	19,721,535.780	0	A
2020-12-31	2020-12-31	18,800,017.170	0	A
2021-01-31	2021-01-31	19,446,307.750	0	A
2021-02-28	2021-02-28	19,973,412.180	0	A
2021-03-31	2021-03-31	19,043,403.310	0	A
2021-04-30	2021-04-30	21,122,491.670	0	A
2021-05-31	2021-05-31	20,478,794.610	0	A
2021-06-30	2021-06-30	20,340,826.070	0	A
2021-07-30		20,522,371.040	0	A
2021-08-30		20,619,804.840	0	A
2021-09-30		20,717,238.630	0	A
2021-10-30		20,811,529.400	0	A

Date	Valid To	Approved Amount	Y	Historical Data
2021-11-30		20,908,963.200	0	A
2021-12-30		21,003,253.970	0	A
2022-01-30		21,100,687.760	0	A
2022-02-28		21,191,835.510	0	A
2022-03-30		21,286,126.280	0	A
2022-04-30		21,383,560.070	0	A
2022-05-30		21,477,850.840	0	A
2022-06-30		21,575,284.630	0	A

Result – Forecast Method: Exponential Smoothing

Date	Valid To	Approved Amount	Y	Historical Data
2014-01-31	2014-01-31	7,915,648.000	0	A
2014-02-28	2014-02-28	8,117,755.340	0	A
2014-03-31	2014-03-31	8,137,351.860	0	A
2014-04-30	2014-04-30	8,613,248.970	0	A
2014-05-31	2014-05-31	9,125,681.200	0	A
2014-06-30	2014-06-30	8,937,232.630	0	A
2014-07-31	2014-07-31	9,509,737.920	0	A
2014-08-31	2014-08-31	9,850,823.910	0	A
2014-09-30	2014-09-30	9,670,498.720	0	A
2014-10-31	2014-10-31	9,723,401.650	0	A
2014-11-30	2014-11-30	9,891,196.600	0	A
2014-12-31	2014-12-31	9,606,154.580	0	A
2015-01-31	2015-01-31	9,771,109.530	0	A
2015-02-28	2015-02-28	10,894,153.580	0	A
2015-03-31	2015-03-31	10,806,004.680	0	A
2015-04-30	2015-04-30	10,900,457.090	0	A
2015-05-31	2015-05-31	11,114,289.780	0	A
2015-06-30	2015-06-30	11,634,529.420	0	A
2015-07-31	2015-07-31	11,746,286.450	0	A
2015-08-31	2015-08-31	11,619,104.800	0	A
2015-09-30	2015-09-30	12,526,260.650	0	A

Date	Valid To	Approved Amount	Y	Historical Data
2015-10-31	2015-10-31	12,539,040.390	0	A
2015-11-30	2015-11-30	12,605,884.700	0	A
2015-12-31	2015-12-31	12,084,553.320	0	A
2016-01-31	2016-01-31	12,857,099.610	0	A
2016-02-29	2016-02-29	13,691,459.800	0	A
2016-03-31	2016-03-31	13,616,689.930	0	A
2016-04-30	2016-04-30	13,868,179.430	0	A
2016-05-31	2016-05-31	13,868,179.430	0	A
2016-06-30	2016-06-30	13,922,138.610	0	A
2016-07-31	2016-07-31	14,331,946.890	0	A
2016-08-31	2016-08-31	14,897,174.960	0	A
2016-09-30	2016-09-30	14,807,930.210	0	A
2016-10-31	2016-10-31	14,786,580.090	0	A
2016-11-30	2016-11-30	14,840,931.360	0	A
2016-12-31	2016-12-31	15,772,728.090	0	A
2017-01-31	2017-01-31	15,104,910.860	0	A
2017-02-28	2017-02-28	15,196,866.190	0	A
2017-03-31	2017-03-31	16,717,905.220	0	A
2017-04-30	2017-04-30	16,320,447.320	0	A
2017-05-31	2017-05-31	16,273,357.880	0	A
2017-06-30	2017-06-30	16,683,750.080	0	A
2017-07-31	2017-07-31	16,335,991.190	0	A
2017-08-31	2017-08-31	17,438,468.070	0	A
2017-09-30	2017-09-30	17,031,429.660	0	A
2017-10-31	2017-10-31	17,914,564.640	0	A
2017-11-30	2017-11-30	16,713,307.780	0	A
2017-12-31	2017-12-31	17,684,150.310	0	A
2018-01-31	2018-01-31	18,088,397.820	0	A
2018-02-28	2018-02-28	17,647,209.960	0	A
2018-03-31	2018-03-31	17,177,858.280	0	A
2018-04-30	2018-04-30	19,117,232.130	0	A
2018-05-31	2018-05-31	19,455,425.620	0	A
2018-06-30	2018-06-30	19,442,605.340	0	A
2018-07-31	2018-07-31	19,500,040.890	0	A

Date	Valid To	Approved Amount	Y	Historical Data
2018-08-31	2018-08-31	18,567,658.220	0	A
2018-09-30	2018-09-30	18,982,459.030	0	A
2018-10-31	2018-10-31	20,718,186.800	0	A
2018-11-30	2018-11-30	19,841,863.110	0	A
2018-12-31	2018-12-31	21,120,797.180	0	A
2019-01-31	2019-01-31	21,477,047.410	0	A
2019-02-28	2019-02-28	19,307,530.800	0	A
2019-03-31	2019-03-31	21,056,419.990	0	A
2019-04-30	2019-04-30	21,564,104.630	0	A
2019-05-31	2019-05-31	21,130,564.960	0	A
2019-06-30	2019-06-30	20,934,401.670	0	A
2019-07-31	2019-07-31	23,245,440.940	0	A
2019-08-31	2019-08-31	22,066,748.240	0	A
2019-09-30	2019-09-30	22,148,297.710	0	A
2019-10-31	2019-10-31	21,527,433.260	0	A
2019-11-30	2019-11-30	22,925,550.220	0	A
2019-12-31	2019-12-31	22,722,256.830	0	A
2020-01-31	2020-01-31	18,849,520.440	0	A
2020-02-29	2020-02-29	18,786,380.170	0	A
2020-03-31	2020-03-31	18,434,899.660	0	A
2020-04-30	2020-04-30	18,564,151.680	0	A
2020-05-31	2020-05-31	18,501,521.020	0	A
2020-06-30	2020-06-30	19,708,359.490	0	A
2020-07-31	2020-07-31	18,829,774.560	0	A
2020-08-31	2020-08-31	18,814,294.240	0	A
2020-09-30	2020-09-30	18,924,925.620	0	A
2020-10-31	2020-10-31	19,971,153.280	0	A
2020-11-30	2020-11-30	19,721,535.780	0	A
2020-12-31	2020-12-31	18,800,017.170	0	A
2021-01-31	2021-01-31	19,446,307.750	0	A
2021-02-28	2021-02-28	19,973,412.180	0	A
2021-03-31	2021-03-31	19,043,403.310	0	A
2021-04-30	2021-04-30	21,122,491.670	0	A
2021-05-31	2021-05-31	20,478,794.610	0	A

Date	Valid To	Approved Amount	Y	Historical Data
2021-06-30	2021-06-30	20,340,826.070	0	A
2021-07-30		20,635,466.290	0	A
2021-08-30		20,807,496.870	0	A
2021-09-30		20,979,527.450	0	A
2021-10-30		21,151,558.020	0	A
2021-11-30		21,323,588.600	0	A
2021-12-30		21,495,619.180	0	A
2022-01-30		21,667,649.750	0	A
2022-02-28		21,839,680.330	0	A
2022-03-30		22,011,710.910	0	A
2022-04-30		22,183,741.490	0	A
2022-05-30		22,355,772.060	0	A
2022-06-30		22,527,802.640	0	A

Related Information

SAP HANA Automated Predictive Library Reference Guide. Version 1902 or later.

1.4.6.1.2 Rule Type: Clustering

The *Machine Learning* function provides rule type *Clustering* to train a clustering model based on input data. The goal of a clustering model is to find underlying structures in the input data, for example segmenting the input data into multiple clusters, where each cluster contains data points that are similar to each other with respect to observed features.

The clustering function runs a k-means algorithm on the input data to determine a partition into clusters. This function then calculates which cluster each data point belongs to. You can choose explicitly which fields (features) of the input data are to be considered by the algorithm. The model that is calculated is saved and stored under a unique model ID for each segment.

Rule Input Fields

You can use the following input fields for configuration:

- *Minimal Number of Cluster*: Specifies the minimal number of clusters
- *Maximal Number of Cluster*: Specifies the maximum number of clusters

- *Clustering Fields*: Specifies the fields (features) according to which the input data is to be clustered
- *Segmented By*: Specifies the fields according to which the whole input data is to be segmented. An independent clustering model is trained for each segment. This means, an independent segmentation into clusters is found. If this list is empty, then the whole input dataset is considered as one segment by default.

The model starts with a cluster number specified by *Minimal Number of Clusters*. It then finds the optimal number of clusters for each input data segment by iterating over different numbers of clusters within the range given by *Minimal Number of Clusters* and *Maximal Number of Clusters* while optimizing certain distance measures within each segment. If the user leaves these two input fields empty, the model will start with default number of 10 clusters.

Rule Output Fields

You can use the following output fields for configuration:

- *Cluster ID*: The field returns the cluster to which an input data point is assigned to by a model. It is an integer by default.
- *Model*: The field returns the unique ID of the trained model for each segment that is being used for predicting. It is a characteristic field type and we recommend you use a length of 30 characters.

1.4.6.1.3 Rule Type: Regression

The *Machine Learning* function provides rule type *Regression* to train and use a regression model for prediction. The idea of a regression model is to define the relationship between input data and target field using training data and the specific functional form learned depends on the choice of model.

The goal of a regression model is to learn to predict an output based on an input set of features. Taking advantage of gradient boosting technique used by SAP HANA automated predictive library (APL), a regression predicts a target field based on influenced fields. Gradient boosting is a machine learning algorithm to find the shortcomings in the previous predictions and combines base learners by sequentially minimizing the difference between the actual and predicted values. It mainly deals with large volumes of data to make a prediction with high prediction power.

Rule Input Fields

You can use the following input fields for configuration:

- *Target Field*: Specifies the field that stores target values of the prediction
- *Influence Fields*: Specifies the model input fields (features) that are used to find the assumed relationship to the target field
- *Segmented By*: Specifies the fields according to which the whole input data is to be segmented. An independent regression model is trained for each segment. This means, an independent segmentation into regression is found. If this list is empty, then the whole input dataset is considered as one segment by default.

- *Order by Fields*: Specifies the fields according to that the segmented datasets are to be sorted.

Rule Output Fields

You can use the following output fields for configuration:

- *Predicted Value*: Specifies a field that stores the predicted values. It must be a key figure of numeric type.
- *Model*: The field returns the unique ID of the trained model for each segment that is being used for predicting. It is a characteristic field type and we recommend you use a length of 30 characters.

1.4.6.1.4 Rule Type: Classification

The *Machine Learning* function provides a rule type *Classification* to train and use a binary or multinomial classification model for prediction. The idea of a classification model is to interpret a relationship between a set of descriptive attributes (features) and a nominal target attribute with two or more than two classes using training data. The specific functional form learned depends on the choice of model.

The goal of a trained classification model is to predict a nominal output based on new input of descriptive attributes. The classification takes advantage of gradient boosting technique empowered by SAP HANA automated predictive library (APL). Gradient boosting is a machine learning algorithm to find the shortcomings in the previous predictions. It combines base learners by sequentially minimizing difference between the actual and predicted values. It mainly deals with large volumes of data to make a prediction with high prediction power.

Rule Input Fields

You can use the following input fields for configuration:

- *Classification Type*: Offers three options – binary, multinomial and autonomous. Default is autonomous. When it is chosen, the algorithm will automatically inspect the classes of the target field in the input data and choose the right model for training.
- *Target Field*: Specifies the field that stores the target attribute of the prediction. The target attribute should be integer-valued.
- *Influence Fields*: Specifies the model input fields (features) that are used to find the assumed relationship to the target field.
- *Segmented By*: Specifies the fields according to which the whole input data is to be segmented. An independent classification model is trained for each segment, for example an independent segmentation into classification is found. If this list is empty, then the whole input dataset is considered as one segment by default.
- *Order by Fields*: Specifies the fields according to which the segmented datasets are to be sorted.

Rule Output Fields

You can use the following output fields for configuration:

- *Predicted Value:* Specifies a field that stores the predicted values
- *Model*: The field returns the unique ID of the trained model for each segment that is being used for predicting. It is acharacteristic field type and we recommend you use a length of 30 characters.

1.4.6.1.5 Rule Type: Recommendation

The *Machine Learning* function provides the rule type *Recommendation* to train and use a recommendation model for prediction. A recommendation model generates recommendations from association rules of the form X => Y. An association rule X => Y means: if an item set X (antecedent) is present, then an attribute Y (consequent) is also present. Two indicators, the support and the confidence, measure the quality of association rules:

- The support of a rule is the number of records verifying the rule. With a rule of the form X =>Y, the support is the count of records containing the item set X and the attribute Y.
- The confidence of a rule is the percentage of records verifying the consequent of the rule among those verifying the antecedent of the rule. With a rule of the form X => Y, the confidence is the count of records containing the item set X and the attribute Y in relation to the count of records containing only the item set X.

Although several rules can lead to the same recommendation, only the metrics corresponding to the rule with the best predictive power (KI) are provided for each recommendation.

The Recommendation rule type creates a model that allows you to perform predictions using the desired input dataset and specifically, to derive recommendations by exploring categorical data, known as items. The function provides clear and useful results, for instance for market basket analysis. It showst the relationships between products or services and immediately suggests appropriate actions.

Once a Recommendation rule type is properly configured and activated, you can run this function and apply recommendation models to the input dataset. Afterwards, the models process input data and determine recommendations for configured user IDs and items in the dataset.

Rule Input Fields

You can use the following input fields for configuration:

- *User ID*: A list of users for which to determine the recommendations.
- *Item*: Fields identifying the user's items in the dataset.
- *Minimum Support*: Minimum value of support to be considered in the association rules. This field requires an integer and the default value is 2.
- *Minimum Confidence*: Minimum value of confidence to be considered in the association rules. It allows you to indicate the minimum confidence that a rule must have to be used as a recommendation. The default value is 0.5.

- *Segmented By*: Specifies the fields according to which the whole input data is to be segmented. An independent recommendation model is trained for each segment. This means an independent segmentation into recommendation is found. If this list is empty, the whole input dataset is considered as one segment by default.

Rule Output Fields

You can use the following output fields for configuration:

- *Recommended Item*: Items recommended by the rules.
- *Recommender Name*: The name of the recommender.
- *Recommendation Rule ID*: The ID of the recommendation rules.
- *Recommendation Score*: The resulting recommendation scores.
- *Model*: Returns the unique ID of the trained model. It is of characteristic type and users are recommended to use a length of 30.

Further Reference

SAP HANA Automated Predictive Library Reference Guide, version 1902 or later.

1.5 How-to Guide

Get an easy start in financial and business modeling by following this How-to Guide.

The following How-to Guide aims to help support business users in their first steps in configuring their own financial models.

1. Modeling and execution of a simple financial and business model
    1. Administration
        1. Check the default settings
           Make sure that default settings have already been defined for Schema, Path, and so on.
        2. Create a team
           Create a team and assign users who will be allowed to later execute the processes and run the simulations.
    2. Modeling
        1. Create an environment
           Set up a non-private environment using the default settings.
        2. Create an information model
           Define fields with master data and hierarchies as well as Model BW functions, which will contain the data during execution.
        3. Create input queries on top of the information model
           Define input-ready queries, which allow data to be entered during execution in a secure way.

4. Create a calculation model
   Define and connect the Join, Derivation, Calculation and Allocation functions, which define the logic of the calculation model.
5. Create report queries on top of the calculation model
   Define read-only queries to visualize and review the results.
6. Define production and simulation process templates with activity templates
   Define the orchestration of the manual and calculation activities.

3. Execution
   1. Deploy production and what-if simulation processes
      Use the prepared templates to deploy a production and simulation process and assign the prepared team.
   2. Execute production process activities
      Run through the production process activities.
   3. Assemble a report, including what-if simulation
      To make the what-if simulation process even more interactive, assemble a report from the simulation process.
   4. Execute the what-if simulation report
      Launch the what-if simulation report, modify data and run the simulation.

Related Information

For more information about financial and business modeling entities, see Financial and Business Modeling Entities [page 7].

For more information about common aspects of SAP Profitability and Performance Management functions, see Modeling Environment [page 48].

Important Disclaimers and Legal Information

Hyperlinks

Some links are classified by an icon and/or a mouseover 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 an 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.

Videos Hosted on External Platforms

Some videos may point to third-party video hosting platforms. SAP cannot guarantee the future availability of videos stored on these platforms. Furthermore, any advertisements or other content hosted on these platforms (for example, suggested videos or by navigating to other videos hosted on the same site), are not within the control or responsibility of SAP.

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 phrasing 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.

Bias-Free Language

SAP supports a culture of diversity and inclusion. Whenever possible, we use unbiased language in our documentation to refer to people of all cultures, ethnicities, genders, and abilities.

www.sap.com/contactsap

THE BEST RUN SAP