Deltek Maconomy Technical Universe Guide

Business Performance Management
— *covering version 2.6*
Copenhagen, 2022

Edited by

Asger Eir

While Deltek has attempted to verify that the information in this document is accurate and complete, some typographical or technical errors may exist. The recipient of this document is solely responsible for all decisions relating to or use of the information provided herein.

The information contained in this publication is effective as of the publication date below and is subject to change without notice.

This edition published July 2022.

Contents

Chapter 1

Introduction

1.1 Objectives

The purpose of this book is to provide technical insights to how the universes of BPM are working technically and how they can be used when writing reports. The book is structured on universes so that each universe is described in it's own chapter.

Note that this book is a living work in progress; not all universes are thoroughly described. It will evolve as BPM is evolved.

Maconomy workflows: For each universe, we describe some general workflows and explain what the effects are in the database. This gives a foundation for explaining how the universe is functioning.

Description of business layers: For each universe, the business layer (measure objects and dimensions objects) are described. The purpose is to give an overview of where to find the relevant and desired objects for reports. As part of the description, measure objects and some central dimensions are described concerning how they are calculated.

Description of data foundation: For each universe, we describe the central parts of the data foundation in order to give an understanding of how data from the Maconomy database are transformed to business values.

Reporting examples: For each universe, we provide examples of how to use the universe in practice.

1.2 Outline

The book is divided into parts; each divided into chapters:

- First chapter (Chapter 1) is this introduction.

- Part I describes the general design principles applied to all BPM universes.
- Part II describes BPM Reporting. Chapter 3 describes the general design principles that specifically applies to BPM Reporting. Each succeeding chapter treats a universe in BPM Reporting.
- Part III describes BPM Analysis. Chapter 26 describes the general design principles that specifically applies to BPM Analysis. Each succeeding chapter treats a universe in BPM Analysis.

1.3 About versions and naming

This book is under constant elaboration and we try to catch up with the latest enhancements to BPM. The book, however, is *not* versioned but is continuously updated with each new version and service pack. The present revision of the book complies with **Maconomy** 2.5.1. For some major functionalities, we state from which version it was introduced, but this is not a consistent approach. Hence, it is important always to check whether the mentioned functionality is indeed available in the version and service pack of BPM that is available on the present installation.

Concerning third party products, we shall assume the latest recommended version that fits with the BPM version.

BPM is an abbrevation for "*Business Performance Management*". However, we shall mainly use "BPM". "BPM Reporting" and "BPM Analysis" are used for referring to the two products of BPM. By "BPM Data Warehouse", we understand the data warehouse and ETL part of BPM Analysis.

For naming of third party products, we shall avoid stating the full names, as it may disturb the reading. Thus, we shall consistently use the following short product names:

BusinessObjects for *SAP BusinessObjects* or *SAP Business Intelligence Platform*.

BO as a short form of *BusinessObjects*.

WebIntelligence for *SAP BusinessObjects WebIntelligence*.

Oracle for the *Oracle* database.

SQL Server for the *Microsoft SQL Server* database.

Part I

General

Chapter 2

General

This chapter describes the general information about the BPM universes. The chapters in Part II describe the universes of BPM Reporting. The chapters in Part III describe the universes of BPM Analysis. Each of the parts introduce by describing the general principles of universes for the product; followed by chapters that in detail describe each universe concerning prerequisites, Maconomy business flows, object definitions, reporting examples and central principles in the design of the data foundation[1].

2.1 The Rôle of a Universe

A universe consists of two things. The first is a collection of database tables and joins between these tables. The purpose of it is to provide a structured foundation for making reporting queries to the database. Typically, a universe is centered on a specific area in the database and selects only tables relevant for reporting on that area. The collection of database tables and joins is called the *Data Foundation*. The second is a collection of calculated *Business Objects*, where each object provides a specific kind of value that can be used in a report[2] or a dashboard. An example is a job number, the registered billing price, financial opening balance, and the remark text of a payment. The purpose of the business objects is two-fold. They serve as an abstraction from the database layer such that—when speaking in terms of the object titles—we talk in the language of the business. Complex table associations and calculations (possibly including technical fields in the database) are abstracted to pure business terms that the report writer or even end-user is familiar with. The other purpose is to encapsulate complex calculations such that these do not need to be done when writing a report; neigther need to be understood by the report writer.

[1] However, not everything is available for all universes.

[2] We shall use the notion of "*report*" and not the notion of "*document*" even though that is—in BusinessObjects—the correct term. However, in most businesses (and maybe especially the businesses that Deltek Maconomy deal with) know what a report is, so we shall speak their language on this point.

2.2 BPM Universes

In BPM Reporting and in BPM Analysis, universes are centered around core Maconomy modules. Each universe contains several hundred objects which make it possible to create a large variety of reports. The universes can also be combined by making report queries to different universes. Thereby, e.g. financial data can be compared to job Cost data, information on opportunities, subscriptions, etc. Any universe can also be combined with the Reporting Structure universe to make it possible to filter and structure on reporting groups of almost any dimension in Maconomy.

The universes in BPM Reporting are made for creating all the daily and straight-forward reports that are needed to show "*live*" data; i.e. data that are up-to-date at any time. When reporting using a BPM Reporting universe, data are fetched directly from the transactional database of Maconomy.

The universes in BPM Analysis are made for creating more complex reports and analysis where complex calculations are needed but where there is not the same demand on having the latest up-to-date transactions included. When reporting using a BPM Analysis universe, data are fetched from the BPM Data Warehouse.

The universes in BPM Reporting and in BPM Analysis may be used in combination to form reports but bare in mind that the two sorts of universes work differently regarding dimensions and other technicalities.

2.3 Dimensions and Facts

There are basically two kinds of objects in the BPM universes: *Dimensions* and *Facts*. Facts are objects providing all the figures that we usually make totals of, take the average of, find the greatest or smallest value of, and so on. Examples are objects displaying the *registered billing price*, *debit* and *credit* amounts from financial transactions, and the *sales estimate* of an opportunity.

Dimensions are objects which are used for characterizing the fact objects. For a given registration in Maconomy's **Job Cost** module, we get a specific registered billing price which we can report on with that corresponding fact object. However, the registration happened on a certain date, within a certain company, associated certain standard dimension values, was done by a certain employee, etc. These data are individual dimensions related to the registration. By characterising the registered billing price with these data, we can restrict on dimensions to find specific registrations, section on dimensions to group registrations with the same dimensional values, or combine with other facts having the same dimensions; e.g. from other universes.

2.4 Dimensional Modelling

There are basically two different principles for designing join structures for databases. The former is called the *Entity-Relationship Model* (ER model). The latter is called *Dimensional Modelling*. In the ER model, we join database tables by their foreign keys or in such a way that we from each table easily can reach the other tables that are somehow associated. The model also seeks to reduce redundancy by breaking up tables to 3NF. Each table may represent a single business concept, but if broken down, the concepts that each table represents may be rather artificial. The compact layout of tables, and the logical relationship defined between them, makes this model suitable for transactional systems where many records are created, deleted and updated, and where we need to keep tables dependent on the changes in various places in the system.

However, the ER model is not suffient when it comes to reporting and data extraction. There are several different reasons for that. Very often report queries involve many tables and as we have decomposed several tables, the number of tables and joins becomes large. Another reason is that the ER model does not ensure that large tables are not joined directly to each other. When updating records in a transactional system, this may be crucial. However, in a reporting situation it can be problematic from a performance point of view.

In a reporting situation, we want to fetch certain figures and associate values by which we can order, group and structure these figures. This brings us to the other design principle: *Dimensional Modelling*. This principle divides database tables into two categories: facts and dimensions. From fact tables, we define the objects showing figures that can be summarized, counted, or otherwise aggregated. An example is the registered number of hours. We call these values or objects *facts*. From dimension tables, we define objects which characterize the facts. An example is the project on which the registrations were made. Another example is the date of each registration. Hence, the dimensions describe the facts (which are merely a number) and we can use the dimensions for structuring and filtering the data. E.g. we can summarize all the registered hours by project, and we can restrict to only see the registrations from the current month.

The tables are joined in a so-called *star schema* where each fact is joined to a number of dimensions; typically on the foreign keys of the fact table and the key fields of the dimension tables. Different fact tables may be joined to the same dimension tables and thereby we can easily report on figures from the different facts and structure and filter the data by the same dimension values.

2.4.1 Multiple facts and queries

Each universe in BPM is designed after the dimensional modelling principle. Some universes contain only one fact tables; other universes contain several fact tables that are combined through common dimension tables. A query can thus contain fact objects from different fact tables. This will generate multiple queries for which the records are merged

by the common dimensions in the report. Likewise, we may combine facts from different universes, and if including alike dimensions from universes, we are able to merge these in the report thereby combining the data from the different universes.

Example 2.1 (Querying different universes) *Assume that we want to report on registered hours and compare these by job to the budgeted hours. From the Job Invoicing universe we could select the objects [`Quantity, Reg.`] and [`Job No.`] to get the registrations. From the Job Budgeting universe we could select the objects [`Quantity`] and [`Job No.`] to get the budget figures. This is done in two different queries. If enabled, the two job number objects from the queries will automatically be merged.*

The measures from the two queries, do not ensure that we only consider time *activities so we should in this case add a filter to each of the two universes:*

```
Activity Type Code Equal to 0
```

We could also have used:

```
Activity Type Equal to Yes
```

However, then we require that the popup value of [`Activity Type`] is in English. The previous version is more generic.

Also add the filter [`Belongs to Latest Approved Revision`] in the job budgeting query, and a prompt on [`Budget Type`]. Else, we will summarize budget figures over all budget types and revisions which is not what we want. ■

But the above can actually be done smarter.

Example 2.2 (Registered and budgeted hours without filter) *Both the Job Invoicing and Job Budgeting universe contain objects specifically for reporting on time activities.*

From the Job Invoicing universe we select the objects [`Quantity, Reg., Time`] and [`Job No.`] to get the registrations. We find this measure in the [`Time Activity`] folder. From the Job Budgeting universe we select the objects [`Quantity, Time`] and [`Job No.`] to get the budget figures. Now, we only need the job budget specific filters. ■

2.5 Facts and Dimensions

Each table in a BPM universe, is either a fact table or a dimension table. On the fact tables, we define the measures and some dimension objects are also defined on the fact tables if it is not convenient to introduce dedicated dimension tables for them. On the dimension tables, we define dimension objecst. Layout-wise, facts are organized to the left in the data foundation; i.e. aligened in the left-most column. First-order dimensions are aligned in the second column from the left. Further levels of dimensions (e.g. dimensions on dimensions) are aligened in additional columns. Figure 2.1–Figure 2.3 illustrate this

layout principle of the data foundation. In some special cases, a universe may have fact tables on different granularity joined together. That is the case for instance in the Job Budgeting universe where we have one fact table for periodic figures in periodic job budgets (most detailed) and another fact table for figures in job budgets in general (less detailed). In such cases, the most detailed are positioned far-left and the less detailed to the right of them.

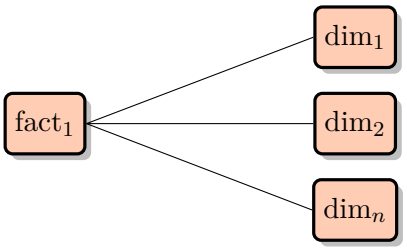


Figure 2.1: Multiple dimensions to single fact

For multiple facts we have:

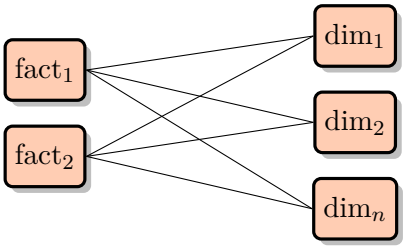


Figure 2.2: Multiple facts to multiple dimensions

And some facts may not have the same dimensions as the other facts

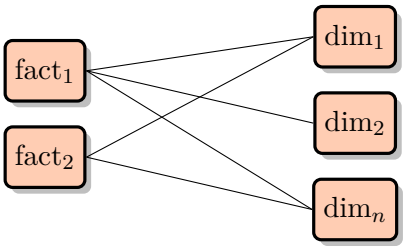


Figure 2.3: Multiple facts to some multiple dimensions

Part II

BPM Reporting

Chapter 3

General Design Principles

This chapter explains the general design principles of the BPM Reporting universes. The succeeding chapters of this part will describe each universe. For each universe, we first give an overall description of what the universe contains and can be used for. Then we describe some typical Maconomy workflows and explain what their impact is in the database.

3.1 Inner joins and outer joins

Most of the joins that are used for associating tables are either inner joins or left-outer joins. The dimensions associated facts are most often mandatory. E.g. standard dimensions (Location, Entity, etc.) as well as company are dimensions that for most facts are always present. An example is that when job entries are created, they always have their **Standard Dimensions** set. The reason is that on fact tables like `JOBENTRY` Maconomy performs **Dimension Derivation** that sets the dimension fields to proper values depepending on the setup in Maconomy. Therefore, we inner join such fact tables to the respective dimension tables of first order. However, there are dimensions that are somehow not required. For these, left-outer joins from the fact table to the dimension table, are defined. In rare situations, full outer joins (cartesian *Joins*) may be used.

Cardinality between facts and their first-order dimensions, as well as between dimension tables and their dimensions (*Snow-Flake Dimensions*, are *many-to-one*. It is a fundamental principle in *Dimensional Modelling*, that for a record in a fact table, we associate only one record in a dimension table; otherwise the dimension is ambiguous.

For dimensions of dimensions (Snowflakes), other rules may apply. Standard dimensions in Maconomy also have dimensions associated but their purpose is to serve as input to the **Dimension Derivation** process. If a dimension on a dimension is not to influence the dimension derivation, the value for the dimension is set to blank in Maconomy. E.g. job may be set to derive a location dimension (which means that the location dimension

on the job is filled out), but not an entity (which means that the entity field is blank. Consequently, the join from such dimension to their dimensions (as in the case with job to location and job to entity) is defined as a join. The cardinality is still *many-to-one*.

3.2 Popups and Date Strings

In Maconomy, popups are enumerated datatypes which in the database have an integer representation. In order to also provide a meaningful textual value when reported on, the so-called *EX View* may be applied. Most tables in the data foundations of the BPM Reporting universes, make use of the EX views. But the views are only used if necessary as they do provide a small performance overhead for looking up the string values for the internal popups. The EX views provide both the external display text of a popup with the same name as the original popup field in the Maconomy table, and the internal popup integer value. The latter field is named with the suffix `PN` which stands for *Popup Number*. The PN fields are suggested for use in joins and internal restrictions as it is often the PN fields that are indexes. The external display text fields of popups are not indexed as they are introduced by the EX view.

Example 3.1 ([Activity Type] as internal and external popup) *Consider the two objects [Activity Type] and [Activity Type Code]. The object [Activity Type] is defined as follows:*

```
BOACTIVITY.ACTIVITYTYPE
```

which is coming from EXACTIVITY.ACTIVITYTYPE as BOACTIVITY is just the access control view applied to EXACTIVITY; see further examplanation in Section 3.3. The field ACTIVITY.ACTIVITYTYPE is in Maconomy an enumerated datatype for the Activity Type popup. This means that in the database it is stored as an integer. The EX view transforms the field to be a string and displays the text values associated with the popup literals:

```
(select
  NAME
from
  POPUPITEM
where
  POPUPTYPENAME='ActivitytypeType'
  and POPUPITEMNUMBER=CUSTOMERBUDGETLINE.ACTIVITYTYPE) ACTIVITYTYPE
```

In universes, the field EXACTIVITY.ACTIVITYTYPE is used for the object [Activity Type] and the field EXACTIVITYTYPE.ACTIVITYTYPEPN is used for the object [Activity Type Code]. ■

The EX views also provide additional date fields. In the Maconomy database, all dates are represented by strings in the format `"YYYY.MM.DD"`: *String Dates*. However, when

selecting dates for restriction in a report, BusinessObjects assume real dates; i.e. values of type Date or DateTime, and the Maconomy database does not provide that. In order to be able to restrict on dates, we need the Maconomy string dates as real dates. The EX views provide these additional versions of the date string fields. The original string dates are, however, recommended for use in joins and internal restriction because these are the indexed fields; the real date fields are not.

3.3 Access Control

On top of the EX views, access control is defined by the so-called *BO views*. All the BO views of BPM follow the same access criteria as defined by default access control in Maconomy.

Some Maconomy tables do not by default define access control. The standard dimension tables are examples. Hence, access control is not applied there. In fact access control is only defined where necessary due to the overhead in obviously has.

3.4 Dates

Most date objects are achieved by joining a fact table with the table view `EXCALENDARDAYPV`. This view is a database view on the performance view `CALENDARDAYPV`. The table includes a record for each date that is part of the **Calendar Day** set up in Maconomy. Furthermore it includes a *null-row* which matches the blank dates in the fact table. Thereby, it is possible to always inner join to the `EXCALENDARDAYPV` instead of making an outer join. This improves performance significantly. An example of a join clause is:

```
EXJOBENTRYJOBINVOICELINEMV.ENTRYDATEDS = EXCALENDARDAY.THEDATEDS
```

The date string fields (DS) are here used as these are the fields that are indexed. The date objects and associated date details are now defined on the `EXCALENDARDAYPV` table like this:

```
EXCALENDARDAYPV.THEDATEDS
```

```
EXCALENDARDAYPV.WEEK
```

etc.

3.5 Prompting objects

Some objects have built-in prompting features. This means that if they are included in a report query, the user will be prompted automatically for entering or selecting a value.

There are different kinds of these objects. Here is a few examples:

- objects which prompt the user for selecting the currency type in which to display an amount. E.g. selecting between displaying the amount in **Company Currency**, **Job Currency** or **Enterprise Currency**. C.f. Section 3.8.1.
- objects which prompt the user for selecting between different date type alternatives. E.g. selecting between displaying the **Entry Date** or the **Finance Entry Date**.
- objects which prompt the user for selecting between different dimensions for slicing data. E.g. selecting between the ten standard dimensions.

When prompts are defined in the report layer (as opposed to prompting objects which are defined in the universes), it is possible to fully control the order in which these prompts should appear. This is not the case with prompting objects defined in the universes. Attempts have been made to define these objects so that they position themselves best possibly by including a value for the *so-called 7. prompt parameter*. However, the success of this is highly dependent on which other objects are included; they could also be tried positioned using that parameter.

R Even though the prompt parameter for positioning prompt order (the *7. prompt parameter*) is called "the 7th", it is actually the 8th. if counting from 1.

3.6 Hierarchies

Most universes contain one or more pre-defined custom hierarchies. The intention with these hierarchies is to make drilling and definition of reports with drilling easy. Furthermore, the purpose is to make naming and principle for hierarchies consistent over the variety of universes in BPM.

The objects chosen for the hierarchies are those that are typically used for drilling in reports. That is:

- concatenated number-and-name or name-and-description objects.
- number/name objects uniquely designating dimensions

3.7 Report-near objects

Most universes contain a dedicated folder containing so-called report-near objects. These are objects which are not really core to the business area of the universe but more relate to syntactical or structural aspects of reports. Examples are:

- Objects used for drilling; typically concatenated number-and-name or name-and-description objects.
- dynamic dimension slicing objects.
- user information objects

- objects for controlling certain syntactical elements like blank strings or other constants.

The folder of report-near objects is named **Report**.

3.8 Naming convensions

All objects in the universes are titled so that indicate what they display in a systematic way. To easier find out what an object displays by looking at it's title, we shall here go through the general naming conventions applied in BPM.

3.8.1 Currency type

For objects displaying amounts in a certain type of currency, we state that type right after the term referring to the kind of amount displayed.

Example 3.2 (Currency type in object titles) *The object [`Registered - Time - Company`] from, the Job Invoicing universe, displays the registered number of hours in the currency of the company* ■

The typical types are:

Company for company currency (also known as the *base* currency)

Customer for currency of the customer

Job for currency of the job

Vendor for currency of the vendor

Enterprise for currency of the enterprise

Opportunity for currency of the opportunity

R Note that the currency type is not the same as the currency. An enterprise could exist of several companies having different currencies. So reporting should always group by the currency in which figures are displayed; unless displaying in enterprise currency for which there can only be one.

As mentioned in Section 3.5, we usually provide both objects displaying in a specific currency type and an object that prompts the user for selecting such a type. The latter objects do not state any currency type.

Example 3.3 (Object prompting for currency type) *In the AR Aging universe we have the three* balance *objects [`Balance, Company`], [`Balance, Customer`] and [`Balance, Enterprise`] stating the balance in different specific currency types. There is also the object [`Balance`] which prompts the user for selecting one of these three*

currency types. It is defined as follows:

```
case @Select(Report\Currency Type)
 when 'Customer'  then @Select(Balance\Balance - Customer)
 when 'Company' then @Select(Balance\Balance - Company)
 when 'Enterprise' then @Select(Balance\Balance - Enterprise)
end
```

The above is not the case in the universes Job Invoicing and Job Budgeting as we here—consistently—have the same currency types available. E.g. cost prices are not available in the currency of the job, as billing prices are. Consequently, most job cost reports are forced to display the amounts in the currency of the company.

3.8.2 Code objects

Far the most objects are displaying their value so that they are meaningful to the user. This is also the case for objects which technically in Maconomy are stored as enumerated datatypes (i.e. **Popup Values**. The database fields coming from the popup fields of Maconomy, are translated into meaningful values by the so-called *EX Views*.

However, there are situations where it actually makes sense to *not* use the objects displaying the meaningful value but instead the internal integer value of the popup item (i.e. coming from the original popup fields):

Internal restrictions If we make internal restrictions in reports or universes, using the original popup fields, we get better performance as these are indexed; the objects displaying the meaningful values are not.

Joins If we make joins on popup fields, this should be done on fields that are indexed. The fields providing the meaningful values by means of the *EX Views* are not indexed.

In order to clearly distinguish the two sorts of objects that are based on popup fields, `Code` is appended to the object title of the objects displaying the internal popup integer.

Chapter 4

Job Invoicing Universe

The Job Invoicing universe provides objects for reporting on job cost registrations, job invoices, and job invoices on account. It is possible to distinguish the different kinds of registrations like registrations on invoiceable and non-invoiceable activities and jobs, tasks and employees, billing and cost prices, and gross margins. It is possible to report on invoiced figures, cost and billing prices. It is also possible to report on the figures derived from both registrations and invoices, like open amount, revenue recognized, WIP, and up/down writings. It is possible to report on invoices on account and slice these according to appropriation and many of the same other dimensions as registrations and invoices can be sliced by.

The measure objects can be grouped and restricted by a variety of dimensions including company, customer, job, activity, task, employee, dates, and the ten standard dimensions.

4.1 Prerequisites

In order to fully benefit from the Job Invoicing universe, Maconomy should be set up such that:

- The system parameter **Use Calendar Day** is enabled and the calendar spans all necessary dates.
- Instance keys for `JOBENTRY` should be enabled.
- Instance keys for `JOBINVOICELINE` should be enabled.
- Instance keys for `JOBINVOICEONACCOUNT` should be enabled.
- Capitalization should be set up in the job parameters and—if capitalized—jobs should either be capitalized to **Cost** or **Billing**.

4.2 Maconomy Workflows

In the following sub-sections we shall look at the common Maconomy workflows that are concerned with the data for which the Job Invoicing universe is made for reporting on. First we shall look at a typical registration and invoicing workflow, and then a workflow for invoicing on account. The two flows provide completely different kinds of data because ordinary invoicing and invoicing on account—in Maconomy—are represented by records in different database tables.

4.2.1 Registration and Invoicing

In Maconomy, approved registrations are stored as records in the `JOBENTRY` table. If the job and activity of the `JOBENTRY` are invoiceable, the entry can be invoiced. The result is records in the `JOBINVOICELINE` table. If we invoice the `JOBENTRY` fully the first time, one record is added in the `JOBINVOICELINE` table. However, ff we invoice the entry in parts (e.g. first the one half of the amount and then later the other half), we get multiple records added to the `JOBINVOICELINE` table.

Consider an employee registering time in a time sheet:

1. In the window **Jobs**, we create a new job with number *1001*.
2. In the window **Job Tasks**, we first run the action **Require Tasks** and then create a task with the name *Business Case*.
3. In the window **Time Sheets**, the user registers 5 hours for each of the five working days for some week.

   This provides a `TIMESHEETHEADER` record and a `TIMESHEETLINE` record.

4. In the window **Approve Time Sheets**, the superior (manager) of the employee approves the registered hours.

   This provides a `JOBENTRY` record.

5. In the window **Invoice Selection**, the `JOBENTRY` is invoiced.

   This provides a `JOBINVOICELINE` record.

From the records, we are able to calculate various figures:

- **Revenue Recognized** which is the revenue amount passed to general ledger.
- **Open Cost** which is the cost price of job entries not yet invoiced.

- **Open Billing** which is the billing price of job entries not yet invoiced.
- **WIP** which is a measure for what registrations that have not yet been invoiced. WIP is calculated in different ways depending on whether capitalization is as cost or billing price (below *je* denotes `JOBENTRY` and *jil* denotes the `JOBINVOICELINE`):

$$\text{WIP} = \begin{cases} \text{Reg. Cost Price}_{je} + \text{Transf. Cost Price}_{je} & \text{if Capitalized at Cost} \\ \quad -\text{Inv. Cost Price} - \text{Transf. Cost Price}_{jil} & \\ \\ \text{Rev. Recogn.}_{je} + \text{Transf. Billing Price}_{je} + \text{Rev. Recogn.}_{jil} & \text{if Capitalized at Billing} \\ \quad -\text{Inv. Billing Price}_{jil} - \text{Transf. Billing Price}_{jil} & \end{cases}$$

4.2.2 Invoicing on Account

Invoices on account on the other hand are not stored as records in `JOBINVOICELINE` but in a dedicated table `JOBINVOICEONACCOUNT`. Usually an invoice on account is followed by a normal invoice. That is for instance the case when prebilling the customer. The prebilling is represented by an invoice on account and the normal invoice is based on the actual work that was done. When the normal invoice is issued, the amount invoiced on account needs to be reduced. This is done by introducing reconciliation records in the table `JOBINVOICEONACCOUNTRECONCILI`. Combining the tables `JOBINVOICEONACCOUNT` and `JOBINVOICEONACCOUNTRECONCILI`, we can determine the amount still to be normally invoiced; i.e. the amount which is still *on account*. This amount is also known as the **NET On Account**.

Consider a prepayment done for a job:

1. In the window **Jobs** create a job with number *1002*.
2. In the window **Job Tasks** run the action **Require Tasks** and create two tasks *Documentation* and *Marketing*.
3. In the window **Job Invoice On Account Selection**, create two lines; one for task *Documentation* with the amount $2500 and one for task *Marketing* with the amount $4600.
4. Approve and print the invoice on account.

   This adds two records to the `JOBINVOICEONACCOUNT` table.
5. In the window **Job Journal**, create a row for task *Documentation* with a total billing price of $1000 and a row for task *Marketing* with a total billing price of $3000.
6. Post the **Job Journal** and invoice the two job entries in the **Invoice Selection**. This adds two records to the `JOBENTRY` table and corresponding two records to the `JOBINVOICELINE` table. It also adds two records to the table `JOBINVOICEONACCOUNTRECONCILI`. On these records it is stated that the reconciled amounts are deduced due to the normal invoice done. The total invoiced on account

was 7100 before TAX and any discounts. The normal invoice was on 4000 so the **NET On Account** is thus 3100.[1]

Invoices on account can also be deducted by making an invoice allocation in which case records in the table `JOBINVOICELINE` are created as well.

4.3 Business Layer

The universe provides measures for reporting on all sorts of registrations and invoices done in the Job Cost module of Maconomy; including invoicing on account. Furthermore, it includes measures for reporting on derived figures like revenue recognized and balances like open amounts and quantities, and WIP. For each of these categories of measures, the business layer has a class containing the measures and dedicated dimensions. In the following sub sections, we shall go through each of these classes and explain the measure objects contained.

The measure objects of the Job Invoicing universe are concentrated on the kind of figures we get out of job cost registration and invoicing. I.e:

- registration figures like registered billing price, registered cost price, revenue recognized calculated at the time of registration, etc.
- invoicing figures like invoiced billing price, invoiced cost price, up/down, etc.
- balances between registered and invoiced like WIP, open cost, open billing, the adjustment to revenue recognized, etc.

In the following we shall present these measure objects of the semantic layer.

4.3.1 Registrations

The dimension and measure objects for reporting on job registration, are located in the [`Registrations`] folder. This folder has two sub-folders:

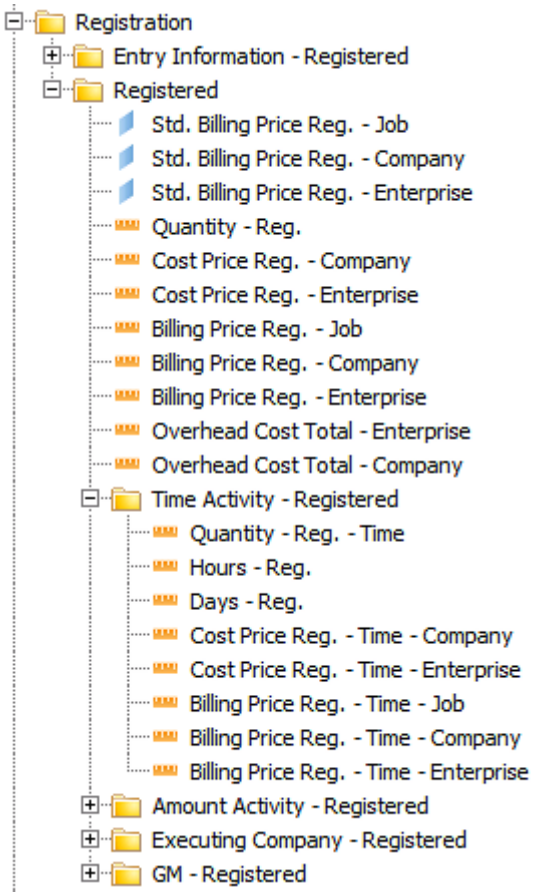
Entry Information which contains dimension objects displaying information like entry number, transaction number and transaction type of the individual job entries.

Registered which primarily contains the measure objects displaying the registered figures.

The latter is what we should concentrate on here. It looks like this:

[1] Note that this is a simplistic scenario where we are not concerned with TAX, invoice discount, etc. so the figures may be different from system setup to system setup.

All the measures are rooted on fields in the JOBENTRY table. However, in fact the objects are defined on the performance view JOBENTRYJOBINVOICELINEPV (see Section 4.5.1.) We utilize the *Aggregate Awareness* so that if possible, the corresponding fields on JOBACTIVITY are used instead. This may be the case if e.g. date, employee or task objects are not included in the query or restrictions.



[Quantity - Reg.]	JOBENTRY.NUMBERREGISTERED
[Cost Price Reg. - Company]	JOBENTRY.COSTPRICEREG
[Cost Price Reg. - Enterprise]	JOBENTRY.COSTPRICEREGENTERPRISE
[Billing Price Reg. - Job]	JOBENTRY.BILLINGPRICEREGCURRENCY
[Billing Price Reg. - Company]	JOBENTRY.BILLINGPRICEREGBASE
[Billing Price Reg. - Enterprise]	JOBENTRY.BILLINGPRICEREGENTERPRISE
[Overhead Cost Total - Enterprise]	JOBENTRY.OVERHEADCOSTTOTALENTERPRISE
[Overhead Cost Total - Company]	JOBENTRY.OVERHEADCOSTTOTALBASE

[Quantity - Reg. - Time]
JOBENTRY.NUMBERREGISTERED
if EXACTIVITY.ACTIVITYTYPEPN $= 0$
0 *otherwise*
[Hours - Reg.]
JOBENTRY.NUMBERREGISTERED
if EXACTIVITY.ACTIVITYTYPEPN $= 0$
and EXJOBHEADER.TIMEREGISTRATIONUNITPN $= 0$
EXJOBHEADER.HOURSPERMANDAY $*$ JOBENTRY.NUMBERREGISTERED
if EXACTIVITY.ACTIVITYTYPEPN $= 0$
and EXJOBHEADER.TIMEREGISTRATIONUNITPN $= 1$
0 *otherwise*
[Days - Reg.]
JOBENTRY.NUMBERREGISTERED
if EXACTIVITY.ACTIVITYTYPEPN $= 1$
and EXJOBHEADER.TIMEREGISTRATIONUNITPN $= 1$
$\frac{\text{JOBENTRY.NUMBERREGISTERED}}{\text{EXJOBHEADER.HOURSPERMANDAY}}$
if EXACTIVITY.ACTIVITYTYPEPN $= 0$
and EXJOBHEADER.TIMEREGISTRATIONUNITPN $= 0$
and EXJOBHEADER.HOURSPERMANDAY ≥ 0.001
0 *otherwise*
[Cost Price Reg. - Time - Company]
JOBENTRY.COSTPRICEREG
if EXACTIVITY.ACTIVITYTYPEPN $= 0$
0 *otherwise*
[Cost Price Reg. - Time - Enterprise]
JOBENTRY.COSTPRICEREGENTERPRISE
if EXACTIVITY.ACTIVITYTYPEPN $= 0$
0 *otherwise*
[Billing Price Reg. - Time - Job]
JOBENTRY.BILLINGPRICEREGCURRENCY
if EXACTIVITY.ACTIVITYTYPEPN $= 0$
0 *otherwise*
[Billing Price Reg. - Time - Company]
JOBENTRY.BILLINGPRICEREGBASE
if EXACTIVITY.ACTIVITYTYPEPN $= 0$
0 *otherwise*
[Billing Price Reg. - Time - Enterprise]
JOBENTRY.BILLINGPRICEREGENTERPRISE
if EXACTIVITY.ACTIVITYTYPEPN $= 0$
0 *otherwise*

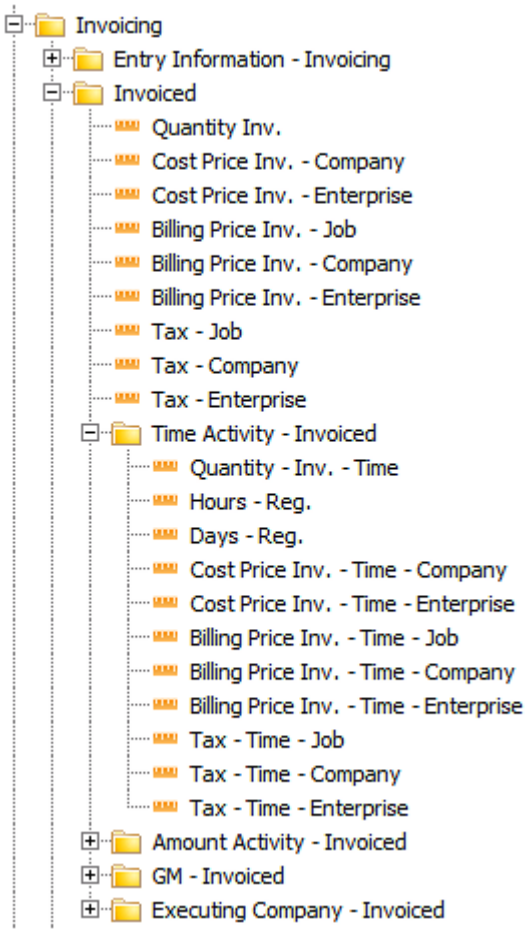
4.3.2 Invoices

The dimension and measure objects for reporting on job invoicing, are located in the [Invoiced] folder. This folder has two sub-folders:

Entry Information which contains dimension objects displaying information like invoice number, entry text, etc. of the individual job invoice lines.

Invoiced which primarily contains the measure objects displaying the invoiced figures.

The latter is what we should concentrate on here. It looks like this:



All the measures are rooted on fields in the JOBINVOICELINE table. As for the registration objects, the objects are defined on the performance view JOBENTRYJOBINVOICELINEPV (see Section 4.5.1.) Also similar as for the registration objects, we utilize the *Aggregate Awareness* to JOBACTIVITY if possible.
Each JOBINVOICELINE record states whether it represents a debited or credited amount. In case of the latter, the sign is reversed. In the following we shall assume debit amounts only.

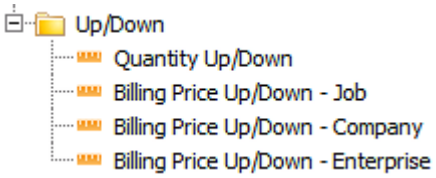
[Quantity Inv.]	JOBINVOICELINE.NUMBEROF
[Cost Price Inv. - Company]	JOBINVOICELINE.COSTPRICETOTAL

[Cost Price Inv. - Enterprise]	JOBINVOICELINE.COSTPRICETOTALENTERPRISE
[Billing Price Inv. - Job]	JOBINVOICELINE.BILLINGPRICETOTALCURRENCY
[Billing Price Inv. - Company]	JOBINVOICELINE.BILLINGPRICETOTALBASE
[Billing Price Inv. - Enterprise]	JOBINVOICELINE.BILLINGPRICETOTALENTERPRISE
[Tax - Job]	JOBINVOICELINE.VATCURRENCY
[Tax - Company]	JOBINVOICELINE.VATBASE
[Tax - Enterprise]	JOBINVOICELINE.VATENTERPRISE

4.3.3 Up/Down Writing

Up/Down writing happens when a registered amount is invoiced with a higher or lower figure.

The objects are defined on fields from the JOBINVOICELINE table. I.e., each invoiced figure represented by a JOBINVOICELINE record states which up/down writing was related to that invoiced figure. However, contrary to invoiced figures, the up/down figures are not controlled by whether the entry is in debit or credit. If no up/down writing was made, the fields are just zero.
The up/down objects do not include the VAT amount. Dedicated objects have been reserved for stating the VAT of the up/down.

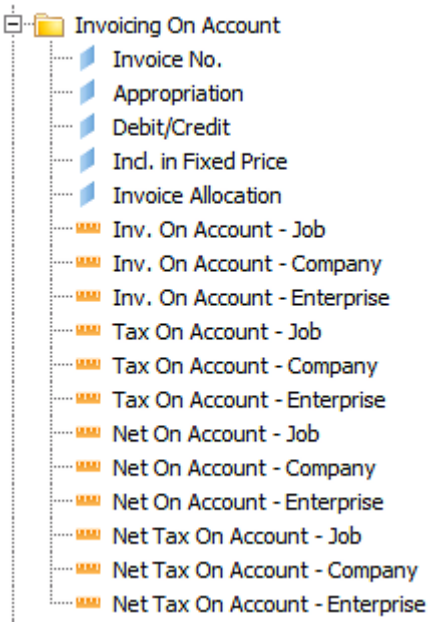


[Quantity Up/Down]	JOBINVOICELINE.NUMBERUPDOWN
[Billing Price Up/Down - Job]	JOBINVOICELINE.BILLINGPRICEUPDOWNCURRENCY
[Billing Price Up/Down - Company]	JOBINVOICELINE.BILLINGPRICEUPDOWNBASE
[Billing Price Up/Down - Enterprise]	JOBINVOICELINE.BILLINGPRICEUPDOWNENTERPRISE

4.3.4 Invoicing On Account

Job invoices on account are in Maconomy stored differently than ordinary invoices. Therefore we report of the invoices on account from the table `JOBINVOICEONACCOUNT`. After a job invoice on account has been issued to the customer, an ordinary invoice is often created. This ordinary invoice is based on the registration that actually took place. As an invoice on account has already been issued we subtract that amount from the ordinary invoice. This also reduces the balance we have on account and this is stored in the `JOBINVOICEONACCOUNTRECONCILI`.

The objects for reporting on the total amount being invoiced on account are the objects named [`Inv. On Account`] of which there are three variants; one for each currency type. The objects for reporting on the balance of what is still invoiced on account and thereby yet to be reconciled against ordinary invoices, are the objects named [`Net On Account`]; again three variants.



[Inv. On Account - Job]	JOBINVOICEONACCOUNT.ITEMSUMCURRENCY
[Inv. On Account - Company]	JOBINVOICEONACCOUNT.ITEMSUMBASE
[Inv. On Account - Enterprise]	JOBINVOICEONACCOUNT.ITEMSUMENTERPRISE
[Tax On Account - Job]	JOBINVOICEONACCOUNT.VATCURRENCY
[Tax On Account - Company]	JOBINVOICEONACCOUNT.VATBASE
[Tax On Account - Enterprise]	JOBINVOICEONACCOUNT.VATENTERPRISE

[Net On Account - Job]	
	JOBINVOICEONACCOUNT.ITEMSUMCURRENCY
	− ∑ JOBINVOICEONACCOUNTRECONCILI.RECONCILEDAMOUNTCURRENCY
[Net On Account - Company]	
	JOBINVOICEONACCOUNT.ITEMSUMCURRENCY
	− ∑ JOBINVOICEONACCOUNT.RECONCILEDAMOUNTBASE
[Net On Account - Enterprise]	
	JOBINVOICEONACCOUNT.ITEMSUMENTERPRISE
	− ∑ JOBINVOICEONACCOUNTRECONCILI.RECONCILEDAMOUNTENTERPRISE
[Net Tax On Account - Job]	
	JOBINVOICEONACCOUNT.VATCURRENCY
	− ∑ JOBINVOICEONACCOUNTRECONCILI.RECONCILEDVATCURRENCY
[Net Tax On Account - Company]	
	JOBINVOICEONACCOUNT.VATBASE
	− ∑ JOBINVOICEONACCOUNTRECONCILI.RECONCILEDVATBASE
[Net Tax On Account - Enterprise]	
	JOBINVOICEONACCOUNT.VATENTERPRISE
	− ∑ JOBINVOICEONACCOUNTRECONCILI.RECONCILEDVATENTERPRISE

In the above, debit and credit needs to be taken into account as well. This means that the above database fields in the expressions are multiplied with -1 if the invoice on account is a credit memo; 1 otherwise.

Note, that amounts that are invoiced on account are not included in calculations of e.g. billing prices or WIP. However, it may make sense to show e.g. Net on Account as an additional column in reports showing these other kinds of amounts. Such amounts should sooner or later appear as added to the existing billed amount so showing the Net on Account can add transparency to the report.

4.3.5 Revenue Recognized

The amount of **Revenue Recognized** in **Job Cost** is first time calculated when registrations are approved; i.e. when job entries are created. This revenue recognized amount is stored on the job entries. When a job entry is invoiced, changes may be done e.g. to the invoiced billing price and that also reflects the revenue recognized amount. This adjustment is stored on the job invoice lines. Thus, to get the total revenue recognized amount, we must add the amount from the job entries and any possible revenue recognized amount changes from the job invoice lines.

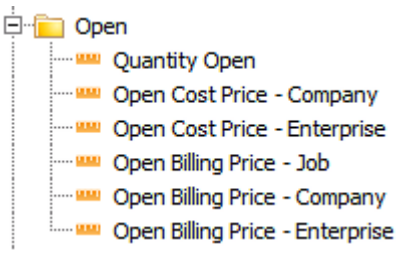
The objects for reporting on the total revenue recognized amount are the objects named [Revenue Recognized]. There is a variant for each currency type. The objects for reporting on the original revenue recognized amount coming entirely from the registrations (i.e. the job entries), are the objects named [Orig. Revenue Recognized]. Subtracting the values of these two kinds of objects, gives the adjustment that was made to the revenue recognized amount as part of invoicing.

[Revenue Recognized - Job] JOBENTRY.REVENUERECOGNIZEDCURRENCY + JOBINVOICELINE.REVENUERECOGNIZEDCURRENCY
[Revenue Recognized - Company] JOBENTRY.REVENUERECOGNIZEDBASE + JOBINVOICELINE.REVENUERECOGNIZEDBASE
[Revenue Recognized - Enterprise] JOBENTRY.REVENUERECOGNIZEDENTERPRISE + JOBINVOICELINE.REVENUERECOGNIZEDENTERPRISE
[Orig. Revenue Recognized - Job] JOBENTRY.REVENUERECOGNIZEDCURRENCY
[Orig. Revenue Recognized - Company] JOBENTRY.REVENUERECOGNIZEDBASE
[Orig. Revenue Recognized - Enterprise] JOBENTRY.REVENUERECOGNIZEDENTERPRISE

4.3.6 Open

Open quantities and amounts are the values on the job entries which are still to be invoiced. We also use the open amounts to calculate *Work in Progress* (WIP); see Section 4.2.1

The quantity object states the quantity that has not yet been invoiced. Combining it with a restriction on activity type, we can either report on the open hours or the open quantity of items. The other objects gives the open cost and the open billing price respectively. Again, notice that the prices are included with different sign depending on whether they are debit or credit (we only state the debit cases below).



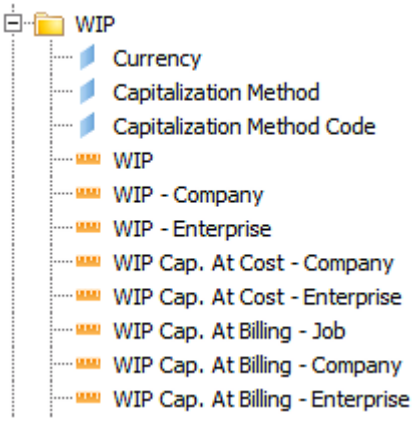
[Quantity Open]		
		JOBENTRY.NUMBERREGISTERED
	+	JOBENTRY.NUMBERTRANSFERREDTO
	+	JOBINVOICELINE.NUMBERUPDOWN
	−	JOBINVOICELINE.NUMBEROF
	−	JOBINVOICELINE.NUMBERMOVED,
		if EXACTIVITY.TOBEINVOICEDPN = 1
		0 *otherwise*
[Open Cost Price - Company]		
		JOBENTRY.COSTPRICEREG
	+	JOBENTRY.COSTPRICETRANSFERREDTO
	−	JOBINVOICELINE.COSTPRICETOTAL
	−	JOBINVOICELINE.COSTPRICETRANSFERREDBASE
[Open Cost Price - Enterprise]		
		JOBENTRY.COSTPRICEREGENTERPRISE
	+	JOBENTRY.COSTPRICETRANSFERREDTOENTERPRISE
	−	JOBINVOICELINE.COSTPRICETOTALENTERPRISE
	−	JOBINVOICELINE.COSTPRICETRANSFERREDENTERPRISE
[Open Billing Price - Job]		
		JOBENTRY.BILLINGPRICEREGCURRENCY
	+	JOBENTRY.BILLINGPRICETRANSFTOCURRENCY
	+	JOBINVOICELINE.BILLINGPRICEUPDOWNCURRENCY
	−	JOBINVOICELINE.BILLINGPRICETOTALCURRENCY
	−	JOBINVOICELINE.BILLINGPRICETRANSFCURRENCY
[Open Billing Price - Company]		
		JOBENTRY.BILLINGPRICEREGBASE
	+	JOBENTRY.BILLINGPRICETRANSFTOBASE
	+	JOBINVOICELINE.BILLINGPRICEUPDOWNBASE
	−	JOBINVOICELINE.BILLINGPRICETOTALBASE
	−	JOBINVOICELINE.BILLINGPRICETRANSFBASE

[Open Billing Price - Enterprise]	
	JOBENTRY.BILLINGPRICEREGENTERPRISE
+	JOBENTRY.BILLINGPRICETRANSFTOENTERPRISE
+	JOBINVOICELINE.BILLINGPRICEUPDOWNENTERPRISE
−	JOBINVOICELINE.BILLINGPRICETOTALENTERPRISE
−	JOBINVOICELINE.BILLINGPRICETRANSFENTERPRISE

4.3.7 WIP

Work in Progress (WIP) is—being a measure on open figures—also an indication of what has been registered but not yet invoiced. However, it takes more parameters into account like job transfer and how the job is capitalized. This is also described in Section 4.2.1.

The object [WIP] shows the work in progress as an amount. It takes the capitalization of the individual job into account and prompts the user for selecting which currency type to display in. The next two WIP objects display in either company or enterprise currency. The additional objects are variants where we display the WIP value as if the job was either capitalized at cost or at billing price.



[WIP - Company]	
	JOBENTRY.COSTPRICEREG
+	JOBENTRY.COSTPRICETRANSFERREDTO
−	JOBINVOICELINE.COSTPRICETOTAL
−	JOBINVOICELINE.COSTPRICETRANSFERREDBASE,
	if capitalized at cost price
	JOBENTRY.ORIGREVENUERECOGNIZEDBASE
+	JOBENTRY.BILLINGPRICETRANSFTOBASE
+	JOBINVOICELINE.REVENUERECOGNIZEDBASE
−	JOBINVOICELINE.BILLINGPRICETOTALBASE
−	JOBINVOICELINE.BILLINGPRICETRANSFBASE
	if capitalized at billing price

[WIP - Enterprise]		
		JOBENTRY.COSTPRICEREGENTERPRISE
	+	JOBENTRY.COSTPRICETRANSFERREDTOENTERPRISE
	−	JOBINVOICELINE.COSTPRICETOTALENTERPRISE
	−	JOBINVOICELINE.COSTPRICETRANSFERREDENTERPRISE
		if capitalized at cost price
		JOBENTRY.ORIGREVENUERECOGNIZEDBASE
	+	JOBENTRY.BILLINGPRICETRANSFTOBASE
	+	JOBINVOICELINE.REVENUERECOGNIZEDENTERPRISE
	−	JOBINVOICELINE.BILLINGPRICETOTALENTERPRISE
	−	JOBINVOICELINE.BILLINGPRICETRANSFENTERPRISE
		if capitalized at billing price
[WIP Cap. At Cost - Company]		
		JOBENTRY.COSTPRICEREG
	+	JOBENTRY.COSTPRICETRANSFERREDTO
	−	JOBINVOICELINE.COSTPRICETOTAL
	−	JOBINVOICELINE.COSTPRICETRANSFERREDBASE
[WIP Cap. At Cost - Enterprise]		
		JOBENTRY.COSTPRICEREGENTERPRISE
	+	JOBENTRY.COSTPRICETRANSFERREDTOENTERPRISE
	−	JOBINVOICELINE.COSTPRICETOTALENTERPRISE
	−	JOBINVOICELINE.COSTPRICETRANSFERREDENTERPRISE
[WIP Cap. At Cost - Job]		
		JOBENTRY.ORIGREVENUERECOGNIZEDCURRENCY
	+	JOBENTRY.BILLINGPRICETRANSFTOCURRENCY
	+	JOBINVOICELINE.REVENUERECOGNIZEDCURRENCY
	−	JOBINVOICELINE.BILLINGPRICETOTALCURRENCY
	−	JOBINVOICELINE.BILLINGPRICETRANSFCURRENCY
[WIP Cap. At Cost - Enterprise]		
		JOBENTRY.COSTPRICEREGENTERPRISE
	+	JOBENTRY.COSTPRICETRANSFERREDTOENTERPRISE
	−	JOBINVOICELINE.COSTPRICETOTALENTERPRISE
	−	JOBINVOICELINE.COSTPRICETRANSFERREDENTERPRISE
[WIP Cap. At Billing - Company]		
		JOBENTRY.ORIGREVENUERECOGNIZEDBASE
	+	JOBENTRY.BILLINGPRICETRANSFTOBASE
	+	JOBINVOICELINE.REVENUERECOGNIZEDBASE
	−	JOBINVOICELINE.BILLINGPRICETOTALBASE
	−	JOBINVOICELINE.BILLINGPRICETRANSFBASE

[WIP Cap. At Billing - Enterprise]	
JOBENTRY.ORIGREVENUERECOGNIZEDBASE	
+	JOBENTRY.BILLINGPRICETRANSFTOBASE
+	JOBINVOICELINE.REVENUERECOGNIZEDENTERPRISE
−	JOBINVOICELINE.BILLINGPRICETOTALENTERPRISE
−	JOBINVOICELINE.BILLINGPRICETRANSFENTERPRISE

When dealing with WIP in a report, there are two aspects that are important to consider. The first is how entries should be restricted concerning dates. The second is how to categorize entries into aging-periods.

Date restrictions in WIP

We have previously seen that the entries to consider in WIP calculation are those of the tables JOBENTRY and JOBINVOICELINE. When we want to calculate WIP, it is always done according to a statement date. That is, we summarize registration figures from the job entries and subtract the summarized invoice figures from the job invoice lines up to a certain date. This means that we usually need to restrict the job entries and job invoice lines by means of a date field on the entries. Another possibility—that we shall consider in Section 4.4.2—is to use an **Aging Principle** and let that categorize the entries into aging periods.

For the registrations, there are two dates that are relevant to use: The ENTRYDATE and FINANCEENTRYDATE; both from JOBENTRY.

For the invoices, the relevant dates are: The ENTRYDATE of the associated JOBENTRY and FINANCEENTRYDATE of the JOBINVOICELINE itself.

The choice of date type (entry date or finance entry date) depends on the strategy and purpose of displaying the WIP figure:

Job Cost related. If we restrict the entries by the ENTRYDATE, we take a *job cost* related approach that is meaningful to most project managers and account managers. The WIP figures in BPM then match the WIP figures displayed i the window **Job Card**.

G/L related. If we restrict the entries by the FINANCEENTRYDATE, we take a *G/L* related approach that is meaningful to finance personnel and accountants. The WIP figures in BPM then matches the WIP figures displayed in the window **Financial Job Card** and they will tie to the **General Ledger**.

R The **G/L** approach can lead to negative WIP figures or negative contributions to WIP. The reason is that a restriction on the job invoice line's FINANCEENTRYDATE may include a job invoice line, while excluding the corresponding job entry. This can happen if an invoice is back dated. This cannot happen if using the ENTRYDATE of the job entries.

R In order to achieve WIP figures that always tie to Maconomy's General Ledger (G/L), restriction on `FINANCEENTRYDATE` must be applied.

Dates when Aging WIP

Aging entries in context of WIP is always done by a date on the `JOBENTRY`. We can either choose to age by `ENTRYDATE` or by `FINANCEENTRYDATE`. The approaches can give two different results as the two dates may differ.

If we age by the `ENTRYDATE` we are concerned with the dates of the approvals of registrations. If we age by the `FINANCEENTRYDATE`, we are more concerned with the dates related to the financial postings of Maconomy.

R WIP can be negative which happens if a job entry lying after the statement date (e.g. a registration from the new month), is invoiced back in time. This gives a negative contribution to the WIP.

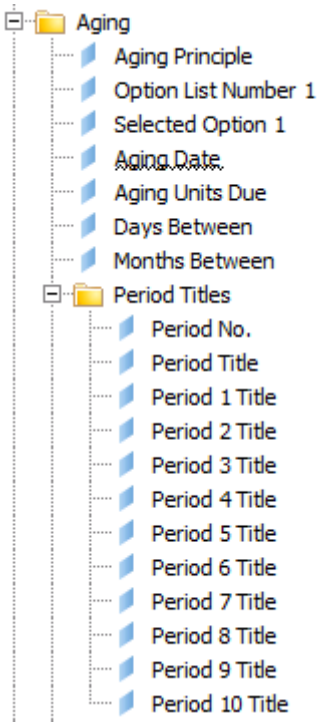
4.3.8 Aging

Work in Progress (WIP) can be aged in order to highlight which uninvoiced amounts that are one month old, two month old, etc. In Maconomy, the aging periods are setup in aging principles. The universe offers two ways of working with aged WIP figures:

- By means of predefined aging period measure objects and dimension titles.
- By means of aging period dimension objects that can be combined with the general [`WIP`] measure object; e.g. in a cross-tab table to provide a flexible WIP Aging report.

The measures and associated dimensions for the first approach, are located in the folder [`Aging`]. The objects for the second approach, are located in the folder [`Aging Principle`].

The measure objects [Period 1]–[Period 10] provide the amounts aged according to the ten aging principles. There are company and enterprise currency specific versions of these measure objects. The aging principle applied is identified by options in an option list. For that, the objects [Option List Number 1] and [Selected Option 1]. The addition dimension objects act as helper objects for categorising the amounts into the periods. The objects [Period No.] and [Period Title] gives the period number and the title of that period for all the periods in the aging principles and for which there are data in the fact table. Similar objects are available in the [Aging Period] but the difference here is that we get the periods regardlessly of whether the fact table contains any data for each period.

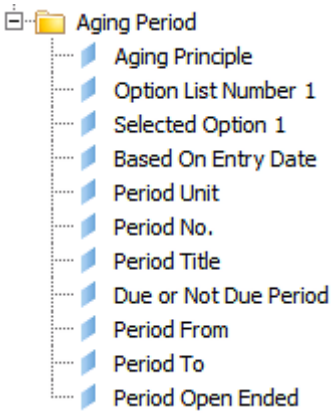


R Note, that the objects [Option List Number 1] and [Selected Option 1] refer to fields that are not mandatory in Maconomy for aging principle. Hence, it is required that the fields are set for the aging principle in order to use the objects for identification and filtering.

4.3.9 Aging Period

The measure objects [Period 1]–[Period 10] mentioned in the previous section have the limitation that the aging reports have to hardcode specific aging columns. From version 2.5.1, new aging dimensions have been introduced in order to query an aging principle separately from registrations and invoices. A separate query to the aging principle that provides the periods that have been setup, can be combined with a query to get the WIP amount, and this can in combination make the foundation in a cross-tab report for a flexible WIP Aging report that automatically adjusts to the aging principle.

The folder [Aging Principle] contains dimension objects for reporting on the aging periods. [Period No.] states the number of the period and [Period Title] are the primary objects. The object [Aging Principle] states the name of the aging principle, whereas the objects [Option List Number 1] and [Selected Option 1] identify the aging principle in the same way as described in the previous section.



4.4 Reporting Examples

4.4.1 Registration in days and hours

From Maconomy version 2.2 (internally 17.0) it is possible to register in whole days as well as in hours. In BPM Reporting this functionality is reflected by two things:

- The normal *quantity* objects for time activities will now display either the number of hours or the number of days; depending on how the job is set up. Additional objects should be used for finding out whether a quantity value for time activities is in hours or in days.
- Additional objects are able to display either in hours only or in days only. This is done by converting days to hours and hours to days, respectively.

Example 4.1 (Sectioning on Job Time Unit) *In the following, we shall build some reports that utilizes this functionality.*

1. *Create a new WebIntelligence document and select the universe* Job Invoicing.
2. *Add the following objects to the query:*

   [Job No.] *(folder [Job])*

   [Job Name] *(folder [Job])*

   [Job Time Unit] *(folder [Job]). This object tells us whether the registered and invoiced time is in days or hours.*

   [Quantity - Reg. - Time] *(folder [Time Activity - Registered])*

   [Quantity - Inv. - Time] *(folder [Time Activity - Invoiced])*
3. *Run the report. We see a table with five columns: The job number and name, the label stating what the time unit for the job is, and the two measures. In Maconomy it is not possible to mix registrations or invoices of time with hours*

on the same job. However, we may want to introduce a sub-total or introduce drilling so that we can see the figures on company or customer level. Therefore it is a good idea to distinguish figures in the two units.

4. *Drag the header of the [`Job Time Unit`] column up above the table to create a section. This gives the necessary distinction and it now makes sense to introduce table totals and drilling.*

labelSomeExample ■

Example 4.2 (Displaying all figures in hours) *Let us look at another example. Assume that we always want to display the registered and invoiced figures in hours. This means that if registrations or invoices are in days, we would need to convert them to number of hours. On the job in Maconomy, it is states how many hours are making one day. The object [`Hours Per Man Day`] shows this rate. However, to build the report, it is not necessary to do the calculations. There are measures that pre-calculate the figures.*

1. *Modify the query to include the objects:*

   [`Job No.`] *(folder [`Job`])*

   [`Job Name`] *(folder [`Job`])*

   [`Hours - Reg.`] *(folder [`Time Activity - Registered`])*

   [`Hours - Inv.`] *(folder [`Time Activity - Invoiced`])*

2. *Modify the table to reflect the query changes*

This report will always display the registered and invoiced quantities for time activities in hours. Similar objects exist for always displaying figures in days. ■

In some reporting situations, we cannot use the unit or number of hours per man day defined on the job. An example is when reporting on the fixed working time for employees. In such cases, the number of hours per man day is defined on the employee revision, and the number of hours per man day is defined on in a system parameter This functionality is defined in the universe Utilization.

4.4.2 Utilizing Aging Principles

Version 2.1 (internally 16.0) introduced the functionality of **Aging Principles**, which makes it easy to create and maintain the aging setup for aging reports in BPM. An aging principle is a named collection of aging periods where each period spans a certain number of days or months and can either be backwards or forwards in time; i.e., a period can be due or not due. Thus, a period in an aging principle could define the period of the last 30 days (30 days due). For WIP aging reports, it is most natural to work with due periods. However, that is completely up to user on how to set it up in Maconomy.

In the following examples, we shall create some different kinds of WIP Aging reports utilizing aging principles. However, before we do that, we need to define an aging principle. In general, for a WIP Aging report, we need the following:

- The period measures and title objects
- Some additional dimensions to slice data by (usually)
- A filter on the aging principle to apply. The aging principle defines the periods in which to place the outstanding registrations. Furthermore, it defines whether to age by the entry date or finance entry date.

Note, that we do not have a filter on any dates. The reason is, that we let it be up to the Aging Period to categorize the entries into aging periods. In the report, we can then choose whether we also want to summarize the measures from these periods and display a total WIP as a column. The period measure objects will indirectly prompt the user to select a statement date which is then the date to which the periods are defined relatively.

Example 4.3 (Simple WIP Aging report) *In this example, we shall create a simple WIP Aging report that prompts the user for a statement date and which aging principle to apply.*

1. *Create a new WebIntelligence document and select the universe* Job Invoicing.
2. *Add the following objects to the query:*

   [Customer No.] *(folder [Company Customer])*

   [Customer Name] *(folder [Company Customer])*

   [Period 1] *(folder [Period Balances])*

   [Period 2] *(folder [Period Balances])*

   [Period 3] *(folder [Period Balances])*

   [Period 4] *(folder [Period Balances])*

   [Period 1 Title] *(folder [Period Titles])*

   [Period 2 Title] *(folder [Period Titles])*

   [Period 3 Title] *(folder [Period Titles])*

   [Period 4 Title] *(folder [Period Titles])*
3. *Add a filter prompt (Equal) on the object [Aging Principle]*
4. *Run the report.*

The report will show the aged WIP for each customer. The WIP figures will be categorized into the four aging periods following the principles defined in the aging principle selected by the user. The figures will be in the currency type chosen by the user.

The filter on the [`Aging Principle`] will make sure that we use one aging principle. Without this filter, we would get duplicated figures because each aged figure would be combined with the matching period existing in each of the aging principles defined in Maconomy. So if we have three aging principles (one for WIP, one for AR and one for AP), we could get our figures tripled. ■

Example 4.4 (Avoiding prompting for an aging principles) *If we wish to always choose a specific aging principle, we can avoid prompting the user for selecting it. This can be achieved by making a filter on the [`Aging Principle`] object restricting it to a specific principle name. However, a more flexible solution is to make restrictions on the option list and option that can be associated the aging principle. Assume that we have set up an option list with the name `"Aging Principles"` and in that created an option called `"WIP Aging"` that this is set up in Maconomy, we just need to do the following adjustments to the report:*

1. *Remove the filter on [`Aging Principle`].*
2. *Add a filter on [`Option List Number 1`] and restrict it to be equal `"Aging Principles"`.*
3. *Add a filter on [`Selected Option 1`] and restrict it to be equal `"WIP Aging"`.*

■

Example 4.5 (Improving performance of WIP Aging) *The performance of a WIP Aging report can be improved significantly by reducing the number of job entries and job invoice lines fetched from the database. In fact most of these are not needed for calculating the WIP figures. Most of the job entries will be matched by corresponding job invoice lines, thereby closing them down and not contributing to WIP. One way to utilize that is to filter off all the entries that are closed before the statement date. I.e., we require that the closing date of an entry is after the statement date.*

1. *Edit the report from Example 4.3 or Example 4.4*
2. *Add a filter on [`Closing Date`] restricting it to `"Greater than"` [`Statement Date`].*
3. *Run the report.*

The filter does not in any way change the result of the report. The [`Closing Date`] object has the value `2200.01.01` if the job entry is not closed at all. If the job entry is

closed, it has the value of the date it was closed. Thereby, if we restrict to the closing date being after the statement date, we filter off all the entries that do not contribute to the WIP anyway. ■

4.5 Data Foundation

This section describes central aspects of the data foundation for the Job Invoicing universe.

4.5.1 Fact Tables

This section describes the fact tables of the data foundation for the Job Invoicing universe.

EXJOBENTRYJOBINVOICELINEPV

This fact table comprises the data for reporting on both registrated and invoiced measures, as well as combined measures like **Work in Progress** and dimensions coming registrations and invoices, or both. The table is stored as a performance view[2] called `JOBENTRYJOBINVOICELINEPV` with the database view `EXJOBENTRYJOBINVOICELINEPV`.

The performance view `JOBENTRYJOBINVOICELINEPV` is comprised by a union of two `select` statements:

- a `select` of fields from the `JOBENTRY` table. Registration measures like registered quantity, and associated dimensions like entry date or job number, is taken from this statement.
- a `select` of fields from the `JOBINVOICELINE` table inner joined to the `JOBENTRY` table. This construct is made so in order to provide the ability to match job entries of the first select, with job invoice lines of the second select. This construct is crucial in order to calculate e.g. **Work in Progress**; see Section 4.2.1.

As the two select statements is a union, they have the exact same fields. The set of fields are divided in parts:

Technical fields which are needed in order to make the performance view able to refresh fast (i.e. incrementally). However, the field `MARKER` can be used to filter on whether to fetch records of the first or the second select in the union. This is utilized a few places in the data foundation of the universe, but it should not be necessary to report on and thus there are no object on this field.

Shared dimensions which are dimension fields that are present both in the first and second select, with a valid value. Some fields are taken from `JOBENTRY` and some

[2] I.e. a *Materialized View* on Oracle and an *Indexed View* on SQL Server.

are taken from `JOBINVOICELINE`. In the second select statements some fields are taken from the `JOBENTRY` table joined to the `JOBINVOICELINE`.

`JOBENTRY` specific fields which are fields taken from `JOBENTRY` in the first select statement only. In the second select statement (invoices) the value is `null` for dimensions and zero for measures. E.g. a daily description is valid only on registrations, whereas a job number is valid both on registrations and invoices.

`JOBINVOICELINE` specific fields which are fields taken from `JOBINVOICELINE` in the second select statement only. In the first select statement (registrations) the value is `null` for dimensions and zero for measures.

For some fields we take the value from just one of the table but for others (especially some measures) the fields provide a calculation where there is both a contribution from job entries and job invoice lines. A good example is the calculation of open billing prices where there is a contribution from job entries in terms of registered and transferred, and a contribution from job invoice lines in terms of invoiced amount and transferred amount. In the latter contribution it is important whether the job invoice line comes from an invoice or credit memo, as this determines whether to subtract or add the measure value.

Below, we state which table, the fields are taken from:

Field	JE	JIL
INSTANCEKEY		
JOBNUMBER		
ENTRYNUMBER		
ACTIVITYNUMBER		
TASKNAME		
EMPLOYEENUMBER		
EXECUTINGCOMPANYNUMBER		
COMPANYNUMBER		
LOCATIONNAME		
PROJECTNAME		
PURPOSENAME		
SPECIFICATION1NAME		
SPECIFICATION2NAME		
SPECIFICATION3NAME		
SPECIFICATION4NAME		
SPECIFICATION5NAME		
SPECIFICATION6NAME		
SPECIFICATION7NAME		
SPECIFICATION8NAME		
SPECIFICATION9NAME		
SPECIFICATION10NAME		
LOCALSPEC1NAME		

LOCALSPEC2NAME
LOCALSPAC3NAME
LOCALSPAC4NAME
LOCALSPAC5NAME
LOCALSPAC6NAME
LOCALSPAC7NAME
LOCALSPAC8NAME
LOCALSPAC9NAME
LOCALSPAC10NAME
JOURNALNUMBER
JOURNALLINENUMBER
TRANSACTIONNUMBER
TRANSACTIONTYPE
TEXT
DIRECTINVOICING
TYPEOFENTRY
ORIGIN
ACRUALENTRY
INCLUDEDINFIXEDPRICE
INVOICEALLOCATION
BASECURRENCYEXECUTINGCOMPANY
INVOICINGJOBNUMBER
REGISTRATIONENTRYDATE
REGISTRATIONENTRYDATEDS
REGISTRATIONFINANCEENTRYDATE
REGISTRATIONFINANCEENTRYDATEDS
FINANCEENTRYDATE
FINANCEENTRYDATEDS
CLOSINGDATE
VATCODE
VATCODE2
VATCODE3
ASSETNUMBER
REMARK
DAILYDESCRIPTION
PAYMENTCUSTOMER
APPROPRIATION
LINENUMBER
DEBITCREDIT
DEBITCREDITPN
INVOICELINETEXT
INVOICELINEORIGIN
INVOICELINEORIGINPN

INVLINEINVOICEALLOCATION	
INVOICENUMBER	
JOBINVOICEALLOCATIONNUMBER	
INVOICEDATE	
INVOICEDATEDS	
REVENUERECOGNIZEDCURRENCY	
REVENUERECOGNIZEDBASE	
REVENUERECOGNIZEDENTERPRISE	
OPENQUANTITY	
OPENCOSTPRICECOMPANY	
OPENCOSTPRICEENTERPRISE	
OPENBILLINGPRICEJOB	
OPENBILLINGPRICECOMPANY	
OPENBILLINGPRICEENTERPRISE	
WIPATBILLINGPRICEJOB	
WIPATBILLINGPRICECOMPANY	
WIPATBILLINGPRICEENTERPRISE	
NUMBERREGISTERED	0
COSTPRISEREG	0
COSTPRICEREGENTERPRISE	0
BILLINGPRICEREGCURRENCY	0
BILLINGPRICEREGBASE	0
BILLINGPRICEREGENTERPRISE	0
COSTPRICEEXECUTINGCOMPANY	0
COSTPRICEEXECCOMPANYENTERPRI	0
ORIGREVENUERECOGNIZEDCURRENC	0
ORIGREVENUERECOGNIZEDB	0
ORIGREVENUERECOGNIZEDENTERPR	0
BILLINGPRICEEXECUTINGCOMPANY	0
BILLINGPRICEEXECCOMPANYENTER	0
STDBILLINGPRICECURRENCY	0
STDBILLINGPRICEBASE	0
STDBILLINGPRICEENTERPRISE	0
OVERHEADCOSTTOTALEXECCOMPBAS	0
OVERHEADCOSTTOTALBASE	0
OVERHEADCOSTTOTALENTERPRISE	0
NUMBEROF	0
COSTPRICETOTAL	0
COSTPRICETOTALENTERPRISE	0
BILLINGPRICETOTALCURRENCY	0
BILLINGPRICETOTALBASE	0
BILLINGPRICETOTALENTERPRISE	0
NUMBERUPDOWN	0

BILLINGPRICEUPDOWNCURRENCY	0	
BILLINGPRICEUPDOWNBASE	0	
BILLINGPRICEUPDOWNENTERPRISE	0	
COSTPRICEINVOICEDEXECBASE	0	
COSTPRICEINVOICEDEXECENTER	0	
VATBASE	0	
VATCURRENCY	0	
VATENTERPRISE	0	

EXJOBACTIVITY

In the Maconomy database, the table `JOBACTIVITY` summarizes certain measure values by some dimensions. The measure values are (here we only list those relevant for job invoicing):

- Quantity registered
- Registered Billing Price
- Registered Cost Price
- Open Quantity
- Open Billing Price
- Open Cost Price
- Quantity invoiced
- Invoiced Billing Price
- Invoiced Cost Price
- Billing Price Up/Down
- Revenue Recognized

Other measure values are also present but the above are the most important. The table summarizes over `JOBENTRY` and `JOBINVOICELINE` by the following dimensions:

- Job
- Activity
- Employee

This means that if we just want to associate these kinds of dimensions, we can utilize the aggregated table `JOBACTIVITY`. However, if we want to e.g. associate with entry date, the `JOBACTIVITY` table cannot be used.

In the data foundation of the Job Invoicing universe, the EX-view of `JOBACTIVITY` is used and joined to its possible dimensions. Measure objects are defined using *Aggregate*

Awareness so that EXJOBACTIVITY is automatically applied if it is possible. If the aggregated table cannot be used, EXJOBENTRYJOBINVOICELINEPV is used instead.

R Notice, that we do not apply the BO view for access control on the objects defined on EXJOBACTIVITY nor for the objects defined onEXJOBENTRYJOBINVOICELINEPV. The reason is that the objects defined on these tables have forced references to the JOBHEADER_D table to which the fact tables are inner joined. This means that we get access control through JOBHEADER_D. The same goes for the fact table JOBINVOICEONACCOUNT_D.

JOBINVOICEONACCOUNT_D

Job invoices on account and their reconciliations are—in Maconomy—stored in the two tables JOBINVOICEONACCOUNT and JOBINVOICEONACCOUNTRECONCILI. The fact table JOBINVOICEONACCOUNT_D comprises these two tables in a union; similar as done in the definition of JOBENTRYJOBINVOICELINEPV. The first select in the union fetches records from EXJOBINVOICEONACCOUNT. The second select fetches records from the result of inner-joining that table with EXJOBINVOICEONACCOUNTRECONCILI. The first select will give us all the job invoices on account. The second select will give us all the reconciliations that have been done against these job invoices on account.

On this structure, we can easily calculate what has been invoiced on account. This is given by certain fields from the first select and these are defined as zero in the second select. An example is the field ITEMSUMBASE which holds the amount invoiced on account in the currency of the company. If we want to calculate **Net On Account**, we need to take what has been invoiced on account from the first select and subtract what has been reconciled against it from the second select. An example is the field NETONACCOUNTJOB which states the net on account in the currency of the job.

An important thing related to dimensions, is that the invoice date in the second select of the fact table, is taken from the field RECONCILIATIONDATE of JOBINVOICEONACCOUNTRE-CONCILI. In versions earlier than 2.3, another solution is used as that field did not exist yet in these versions.

Chapter 5

Job Budgeting Universe

The Job Budgeting universe provides objects for reporting on job budgets and periodic job budgets. It is possible to slice and restrict on the different budget types and workflow stages of budgets like revisions and approvals. It is also possible to report on budgets on the summarized level and on the individual levels where budgets are broken down on activities, tasks and employees. It is possible in this context to report on the different task levels that can be set up in Maconomy. The universe contains measures for reporting on budgeted quantities (hours and units), budgeted cost and billing prices as well as gross margins. When reporting on budgets (either as totals or the individual lines) amounts from possible purchases, can be included to give a picture of the commited amounts for the budget. For periodic job budgets, dimensions like fiscal periods, months, years, etc. can be associated the periodic figures.

The measure objects can be grouped and restricted by a variety of dimensions including company, customer, job, employee, dates, and the standard dimensions.

5.1 Prerequisites

In order to fully benefit from the Job Budgeting universe, Maconomy should be set up such that:

- The system parameter **Use Calendar Day** is enabled and the calendar spans all necessary dates.

5.2 Maconomy Workflows

When creating job budgets in Maconomy the most central kind of records are the **Job Budget Lines**. These are stored in the table `JOBBUDGETLINE`.

Each job budget has a header record in the table `JOBBUDGET` and if revisions have been made of the job budgets, there will be records in the table `JOBBUDGETREVISION` as well.

The records in the JOBBUDGETREVISION are similar to the record in the JOBBUDGET as they work as header records for a collection of job budget lines. However, the records in JOBBUDGETREVISION represent the different revisions of the budget. Such revisions can either be made by simply running the action **New Revision** or by approving and reopening the budget. The records in the JOBBUDGET always represent the latest revision. There is always such a record for a job budget - even though it has no lines. However, records in JOBBUDGETREVISION are not created until revisions are actually made. If the latest revision is approved, the record in JOBBUDGETLINE and the latest revision in the JOBBUDGETREVISION, represent the same budget.

Each budget can appear on different **Job Budget Types**. For one job we can have a **Baseline Budget**, a **Working Budget**, a **Planning Budget**, etc. In the database this is represented as three records in the table JOBBUDGET and three collections of records in the table JOBBUDGETLINE. If there are approved revisions, there will be three records in JOBBUDGETREVISION for each revision, and for each of these we will similarly have collections of records in JOBBUDGETLINE.

5.2.1 Job Budgets, approval and revisions

In the following scenario, we shall create a job budget and carry out approval and create revisions.

1. Create a new job in the window **Jobs**.
2. In the window **Job Tasks** define 6 tasks; e.g. some on time activities and others on amount activities.
3. In the **Job Budgets** windows locate the job just created.
4. In the lower panel, create 6 rows; one for each task. Assign quantities to the tasks.

   This provides 6 *records in the JOBBUDGETLINE. There is one record in the table JOBBUDGET and none in the table JOBBUDGETREVISION*
5. Approve the job budget.

   This adds a record to the table JOBBUDGETREVISION. No new job budget lines are created.
6. Reopen the job budget. This will automatically create a new revision.

   This will mark the record in the JOBBUDGETREVISION as not the latest revision. It will also add a new record to that table, mark the record in the JOBBUDGET as the second revision and create copies of all the job budget lines that are each marked with the second revision. The total number of job budget lines is now 12.
7. Remove one of the job budget lines.

   This reduces the total number of job budget lines to 11.

8. Create a new revision.

   This adds 5 *additional lines to the collection of* `JOBBUDGETLINE` *records and we now have* 3 *revisions in total: one with* 6 *lines and two with* 5 *lines*

5.2.2 Work Breakdown Structure

The lines in a job budget can be hierarchically organized, in the sense that some lines can have the type `"Sum/Text"` and appear as *"parent"* to other lines. Thereby, we express that some work is broken down into sub-work or phases. Consider a variant of the previous scenario:

1. Create a new job in the window **Jobs**.
2. In the window **Job Tasks** define 6 tasks; e.g. some on time activities and others on amount activities.
3. In the **Job Budgets** windows locate the job just created.
4. In the lower panel, create 6 rows; one for each task. Assign quantities to the tasks.
5. Insert an additional line in the top of the lower panel. The line should have the type `"Sum/Text"`.
6. Mark the 6 lines below it and click `Indent`.

   The six lines will now be indented to the right. We get a small button which—when clicked—collapses the six lines so that it is only the `"Sum/Text"` *line that is visible. I.e., that line defines a group of job budget lines. The total number of job budget line records is* 7 *as the* `'Sum/Text"` *line is also stored in the database. Usually, we do not want to report on such summary lines as we in WebIntelligence have more sophisticated means for grouping and defining summaries of job budget lines.*

5.2.3 Periodizing Job Budgets

Job budgets are in a sense time-less even though it is possible to state a start and an end date for the individual lines. For some kinds of jobs, it makes sense to distribute the total number of estimated hours over a selection of periods; thereby, assigning budgeted amounts to these periods. This can be done by periodizing a job budget. Consider the following scenario:

1. Assume one of the job budgets from the previous scenarios.
2. In the window **Periodic Job Budgets** locate the job in question.
3. In the upper panel, choose a period and a unit to divide that period by; e.g. monthly.
4. For each job budget line in the lower panel, distribute the figures to the desired months. There are fields on each line for the periods defined in the upper panel.

For each period where a non-zero value is assigned, a record in created in the table `JOBBUDGETLINEPERIOD`.

5.3 Business Layer

The universe provides measures for reporting on two different sorts of budgets in the **Job Cost** module of Maconomy:

Job Budgets which typically state budgeted quantities, cost and billing prices for tasks of a job.

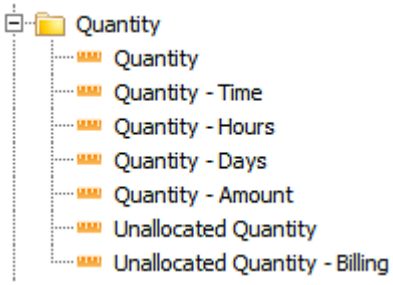
Periodic Job Budgets which distribute figures of a job budget across selected periods.

This distinction reflects in the folder structure of the business layer. There is a folder *Budgeted* which contains objects related to reporting on job budgets, and there is a folder *Budgeted by Period* which contains objects related to reporting on periodic job budgets.

5.3.1 Quantity

The dimension and measure objects for reporting on budgeted quantities from job budgets, are located in the `Quantity` folder.

There are four quantity objects which slice data on activity type. The object [`Quantity - Time`] provides the budgeted quantity on time activities, despite whether the job is set up for registering in days or hours. The object [`Quantity - Hours`] converts that number so that it always displays in a number of hours. The object [`Quantity - Days`] does similarly; just for days. The object [`Quantity - Amount`] displays the budgeted quantity for amount activities, and the object [`Quantity`] simply displays the quantity disregarding activity type. The two `Unallocated` objects display the quantity that is not allocated to any periods in the periodic job budget.



[Quantity]	JOBBUDGETLINE.NUMBEROF
[Quantity - Time]	JOBBUDGETLINE.NUMBEROF *if* EXJOBBUDGETLINE.ACTIVITYTYPEPN = 0

[Quantity - Hours]	JOBBUDGETLINE.NUMBEROF *if* EXJOBBUDGETLINE.ACTIVITYTYPEPN = 0 *and* EXJOBHEADER.TIMEREGISTRATIONUNITPN = 0 JOBBUDGETLINE.NUMBEROF * JOBHEADER.HOURSPERMANDAY *if* EXJOBBUDGETLINE.ACTIVITYTYPEPN = 0 *and* EXJOBHEADER.TIMEREGISTRATIONUNITPN = 1
[Quantity - Days]	JOBBUDGETLINE.NUMBEROF *if* EXJOBBUDGETLINE.ACTIVITYTYPEPN = 0 *and* EXJOBHEADER.TIMEREGISTRATIONUNITPN = 1 $\frac{\text{JOBBUDGETLINE.NUMBEROF}}{\text{JOBHEADER.HOURSPERMANDAY}}$ *if* EXJOBBUDGETLINE.ACTIVITYTYPEPN = 0 *and* EXJOBHEADER.TIMEREGISTRATIONUNITPN = 0
[Quantity - Amount]	JOBBUDGETLINE.NUMBEROF *if* EXJOBBUDGETLINE.ACTIVITYTYPEPN = 1
[Unallocated Quantity]	JOBBUDGETLINE.UNALLOCATEDNUMBER
[Unallocated Quantity - Billing]	JOBBUDGETLINE.UNALLOCATEDSALESNUMBER

R The old-fashioned way of defining *sum*-lines for job budget lines, where the activity number is set to *, is not used in BPM Reporting. Only job budget lines having a proper activity are considered.

R Job budget lines that are sum-lines (i.e. have sub-ordinate lines) are *not* included in BPM-reporting. Measures of individual value-carrying lines will automatically be summarized when making reports.

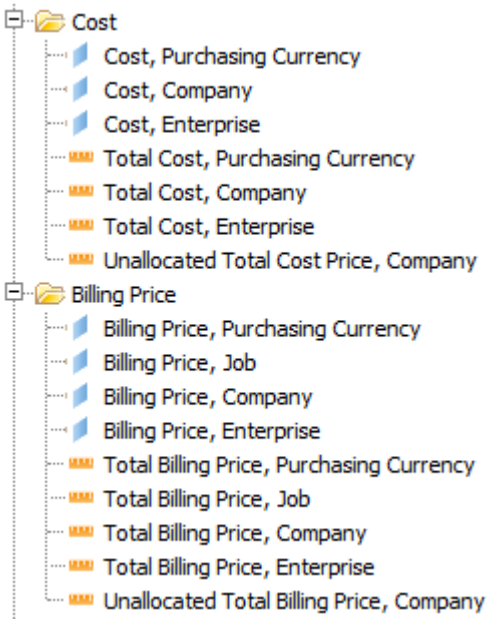
5.3.2 Cost and Billing Prices

The dimension and measure objects for reporting on the budgeted cost and billing prices of job budgets, are located in the [Quantity] folder.

For both cost and billing prices there are basically two sorts of objects. The Cost dimension objects state the unit cost price from a job budget line. The Total Cost measure objects state the total cost budgeted; calculated as the unit cost price times the quantity. Billing Price dimension objects state the unit billing price from a job budget

line. The `Total Billing Price` measure objects state the total billing price budgeted; calculated as the unit billing price times the quantity.

Cost prices are available in the currency of the purchase, the company, and the enterprise. Billing prices are available—in addition—in the currency of the job. There is also an object displaying the unallocated total cost and an object displaying the unallocated total billing, respectively.



[Total Cost - Purchasing Currency] JOBBUDGETLINE.BILLINGPRICETOTALPURCHASINGC
[Total Cost - Company] JOBBUDGETLINE.COSTPRICETOTAL
[Total Cost - Enterprise] JOBBUDGETLINE.COSTPRICETOTALENTERPRISE
[Unallocated Total Cost Price - Company] JOBBUDGETLINE.UNALLOCATEDCOSTPRICETOTAL
[Total Billing Price - Purchasing Currency] JOBBUDGETLINE.COSTPRICETOTALPURCHASINGCURR
[Total Billing Price - Job] JOBBUDGETLINE.BILLINGPRICETOTALCURRENCY
[Total Billing Price - Company] JOBBUDGETLINE.BILLINGPRICETOTALBASE
[Total Billing Price - Enterprise] JOBBUDGETLINE.BILLINGPRICETOTALENTERPRISE
[Unallocated Total Billing Price - Company] JOBBUDGETLINE.UNALLOCATEDBILLINGPRICETOTAL

5.3.3 Gross Margins

The measure objects for reporting on **Gross Margins** of job budgets, are located in the [Gross Margin] folder.

By a **Gross Margin** for budgets, we understand the difference between the budgeted billing price and the budgeted cost price.

By a **Gross Margin %**, we understand the gross margin divided by the total cost price budgeted.

R There are many different understandings of how a gross margin should be calculated. Other ways calculate it as the difference between the budgeted billing (or cost) less the actual figures of e.g. registrations. Some calculate the percentage by dividing with the billing price, etc. The present objects described here, are the most common understandings of job budget related gross margins in Maconomy.

The [Gross Margin] provides the difference between billing and cost budgeted. The [Gross Margin %] divides that number with the total billing price. The [Markup %] is similar; it just divides with the total cost price.

[Gross Margin]	JOBBUDGETLINE.BILLINGPRICETOTALBASE − JOBBUDGETLINE.COSTPRICETOTAL
[Gross Margin %]	$\frac{\text{[Gross Margin]}}{\text{JOBBUDGETLINE.BILLINGPRICETOTALBASE}}$
[Markup %]	$\frac{\text{[Gross Margin]}}{\text{JOBBUDGETLINE.COSTPRICETOTAL}}$

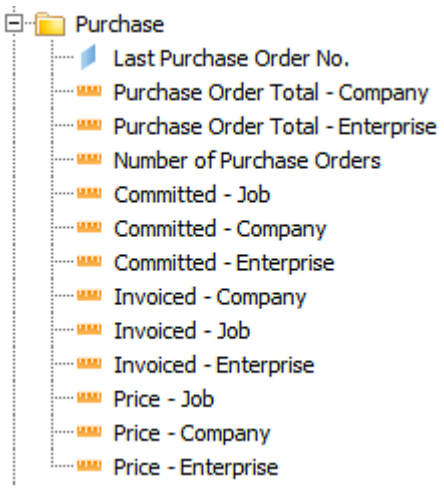
5.3.4 Purchases

The dimension and measure objects for reporting on purchases made in context of job budgets, are located in the [Purchase] folder.

On a job budget, purchases can be associated either directly when operating in the window **Job Budgets** or from the window **Purchase Orders** referencing a job (but not a budget type). BPM Reporting captures both kinds of purchases. Each purchase order has a total price that can be reported on and an invoiced amount which is the amount registered on vendor invoices for the purchase. Furthermore, the notion of **Comitted** is

the total price of the purchase order minus the invoiced amount. Thus, it is relevant to include whenever reporting on budgeted figures, if purchases may be included. Typically, the total price of the purchase order appears as budgeted cost. To outline what has been invoiced and what is left, the comitted figure can be included in the report in addition to the budgeted cost.

There are four kinds of objects. The `Purchase Order Total` objects provide the total amount of the purchase order price in company or enterprise currency. The `Invoiced` objects provide the ability to report on what has been invoiced in the job, company or enterprise currency. The `Comitted` objects provide the ability to report on what is left to be invoiced (the purchase price minus the invoiced price).



[Purchase Order Total, Company]	JOBBUDGETLINE.PURCHASEORDERSUMBASE
[Purchase Order Total, Enterprise]	JOB BUDGETLINE.PURCHASEORDERSUMENTERPRISE
[Number of Purchase Orders]	JOBBUDGETLINE.NUMBEROFPURCHASEORDERS
[Comitted - Job]	PURCHASEORDERLINE.PRICECURRENCY −PURCHASEORDERLINE.INVOICEDCURRENCY *if* PURCHASEORDERLINE.PRICECURRENCY −PURCHASEORDERLINE.INVOICEDCURRENCY > 0 0 *otherwise*
[Comitted - Company]	PURCHASEORDERLINE.PRICEBASE −PURCHASEORDERLINE.INVOICEDBASE *if* PURCHASEORDERLINE.PRICEBASE −PURCHASEORDERLINE.INVOICEDBASE > 0 0 *otherwise*

[Comitted - Enterprise]	PURCHASEORDERLINE.PRICEENTERPRISE −PURCHASEORDERLINE.INVOICEDENTERPRISE *if* PURCHASEORDERLINE.PRICEENTERPRISE −PURCHASEORDERLINE.INVOICEDENTERPRISE > 0 0 *otherwise*
[Invoiced - Job]	PURCHASEORDERLINE.INVOICEDCURRENCY
[Invoiced - Company]	PURCHASEORDERLINE.INVOICEDBASE
[Invoiced - Enterprise]	PURCHASEORDERLINE.INVOICEDENTERPRISE
[Price - Job]	PURCHASEORDERLINE.PRICECURRENCY
[Price - Company]	PURCHASEORDERLINE.PRICEBASE
[Price - Enterprise]	PURCHASEORDERLINE.PRICEENTERPRISE

5.3.5 Periodic Job Budgets

The dimension and measures objects for reporting on periodic job budgets, are located in the [Budgeted By Period] folder. The objects in the folder are organized into the following six sub-folders:

Period This folder contains dimension objects that display the periods, the start and end of periods, and similar related to fiscal years and period.

Quantity (By Period) This folder contains the objects for reporting on the quantity budgeted: for time activities in general, in hours, or days, and for amount activities.

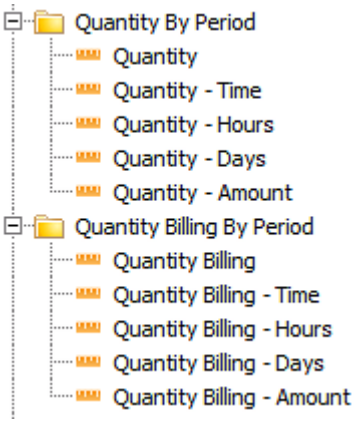
Quantity, Billing (By Period) This folder contains the objects for reporting on the sales quantity budgeted: for time activities in general, in hours or days, and for amount activities.

Cost Price (By Period) This folder contains a dimension object for displaying the unit cost price on the job budget line, and a measure object for displaying the total cost price by period from the periodic job budget. The figures are in the currency of the company.

Billing Price (By Period) This folder contains two dimension objects for displaying the unit billing price on the job budget line in the currency of the company or the job. It also contains two measure objects for displaying the total billing price by period from the periodic job budget; displayed in the currency of the company or the job.

Gross Margin (By Period) This folder contains two objects for reporting on the gross margin and the gross margin percentage of periodic job budgets.

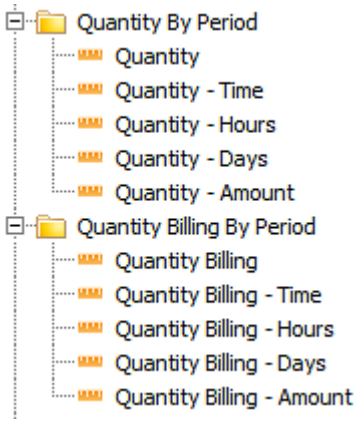
The [Quantity] object displays the quantity budgeted disregarding the activity type and whether time for the job is in days and hours. The [Quantity - Time] object restricts to time activities but does not distinguish between time in days or hours. The [Quantity - Hours] and the [Quantity, Days] objects restrict to time activities and always display in hours and days, respectively. The [Quantity - Amount] restricts to quantity budgeted on amount activities.



[Quantity]	JOBBUDGETLINEPERIOD.NUMBEROF
[Quantity - Time]	JOBBUDGETLINEPERIOD.NUMBEROF *if* EXJOBBUDGETLINEPERIOD.ACTIVITYTYPEPN = 0
[Quantity - Hours]	JOBBUDGETLINEPERIOD.NUMBEROF *if* EXJOBBUDGETLINEPERIOD.ACTIVITYTYPEPN = 0 *and*EXJOBHEADER.TIMEREGISTRATIONUNITPN = 0
[Quantity - Days]	JOBBUDGETLINEPERIOD.NUMBEROF *if* EXJOBBUDGETLINEPERIOD.ACTIVITYTYPEPN = 0 *and*EXJOBHEADER.TIMEREGISTRATIONUNITPN = 1 $\frac{\text{JOBBUDGETLINEPERIOD.NUMBEROF}}{\text{JOBHEADER.HOURSPERMANDAY}}$ *if* EXJOBBUDGETLINEPERIOD.ACTIVITYTYPEPN = 0 *and*EXJOBHEADER.TIMEREGISTRATIONUNITPN = 0
[Quantity - Amount]	JOBBUDGETLINEPERIOD.NUMBEROF *if* EXJOBBUDGETLINEPERIOD.ACTIVITYTYPEPN = 1
[Quantity Billing]	JOBBUDGETLINEPERIOD.SALESNUMBEROF
[Quantity Billing - Time]	JOBBUDGETLINEPERIOD.SALESNUMBEROF *if* EXJOBBUDGETLINEPERIOD.ACTIVITYTYPEPN = 0

[Quantity Billing - Hours] JOBBUDGETLINEPERIOD.SALESNUMBEROF *if* EXJOBBUDGETLINEPERIOD.ACTIVITYTYPEPN = 0 *and*EXJOBHEADER.TIMEREGISTRATIONUNITPN = 0
[Quantity Billing - Days] JOBBUDGETLINEPERIOD.SALESNUMBEROF *if* EXJOBBUDGETLINEPERIOD.ACTIVITYTYPEPN = 0 *and*EXJOBHEADER.TIMEREGISTRATIONUNITPN = 1 $\frac{\text{JOBBUDGETLINEPERIOD.SALESNUMBEROF}}{\text{JOBHEADER.HOURSPERMANDAY}}$ *if* EXJOBBUDGETLINEPERIOD.ACTIVITYTYPEPN = 0 *and*EXJOBHEADER.TIMEREGISTRATIONUNITPN = 0
[Quantity Billing - Amount] JOBBUDGETLINEPERIOD.SALESNUMBEROF *if* EXJOBBUDGETLINEPERIOD.ACTIVITYTYPEPN = 1

Cost price and billing price objects are available in the different currency types. For reporting on the individual cost and billing prices stored on the individual job budget line periods, the dimension objects can be used. Otherwise the measure objects should be used as usual.



[Cost Price - Company] JOBBUDGETLINEPERIOD.COSTPRICE
[Total Cost - Company] JOBBUDGETLINEPERIOD.COSTPRICETOTAL
[Billing Price - Job] JOBBUDGETLINEPERIOD.BILLINGPRICECURRENCY
[Billing Price - Company] JOBBUDGETLINEPERIOD.BILLINGPRICEBASE
[Total Billing Price - Job] JOBBUDGETLINEPERIOD.BILLINGPRICETOTALCURRENCY
[Total Billing Price - Company] JOBBUDGETLINEPERIOD.BILLINGPRICETOTALBASE

[Gross Margin]	JOBBUDGETLINEPERIOD.BILLINGPRICEBASE −JOBBUDGETLINEPERIOD.COSTPRICEBASE
[Gross Margin %]	$\frac{\text{[Gross Margin]}}{\text{JOBBUDGETLINEPERIOD.COSTPRICEBASE}}$ *if* JOBBUDGETLINEPERIOD.COSTPRICEBASE $\neq 0$ 0 *otherwise*

5.4 Reporting Examples

5.4.1 Matching registered hours against budgeted

The most important use of objects from the Job Budgeting universe is to match job figures (registered and invoiced) against budgeted figures. In the following we shall illustrate different approaches for doing so.

Example 5.1 (Matching registered hours against budgeted) *In this example, we shall create a simple report which compares registered hours against the corresponding budgeted hours.*

1. *Create a new WebIntelligence document and select the universe Job Invoicing.*
2. *Add the following objects to the query:*

   [Job No.] *(folder [Job])*

   [Job Name] *(folder [Job])*

   [Job Time Unit] *(folder [Job]). This object tells us whether the registered and invoiced time is in days or hours.*

   [Quantity - Reg. - Time] *(folder [Time Activity (Registered)])*
3. *Add a second query and select the universe Job Budgeting the following objects:*

   [Job No.] *(folder [Job])*

   [Job Name] *(folder [Job])*

   [Job Time Unit] *(folder [Job]). This object tells us whether the registered and invoiced time is in days or hours.*

   [Quantity - Time] *(folder [Quantity])*
4. *Add the filter [Belongs To Latest Approved Revision] (folder [Belongs To])*

5. *Merge the following dimension objects from the two queries: [Job No.], [Job Name] and [Job Time Unit].*
6. *In the report, create a table with the merged dimensions, and the measure objects from the two queries.*
7. *Drag the merged [Job Time Unit] object to create a section.*
8. *Refresh the report.*

R *It is very important to make this section on the object [Job Time Unit]; else we may get quantities in hours mixed with quantities in days.*

■

Example 5.2 (Matching registered prices against budgeted) *In this example, we shall create a simple report which compares registered cost and billing prices against the corresponding budgeted figures.*

1. *Create a new WebIntelligence document and select the universe* Job Invoicing.
2. *Add the following objects to the query:*

   [Company No.] *(folder [Company])*

   [Company Currency] *(folder [Currency])*

   [Job No.] *(folder [Job])*

   [Job Name] *(folder [Job])*

   [Cost Price Inv. - Company] *(folder [Invoiced])*

   [Billing Price Inv. - Company] *(folder [Invoiced])*
3. *Add a second query and select the universe* Job Budgeting *the following objects:*

   [Company No.] *(folder [Company])*

   [Company Currency] *(folder [Currency])*

   [Job No.] *(folder [Job])*

   [Job Name] *(folder [Job])*

   [Total Billing Price - Company] *(folder [Billing Price])*

   [Total Cost Price - Company] *(folder [Cost Price])*
4. *Add the filter [Belongs To Latest Approved Revision] (folder [Belongs To])*

5. *Merge the following dimension objects from the two queries: [Company No.], [Company Currency], [Job No.], and [Job Name].*
6. *In the report, create a table with the merged dimensions and the measure object from the two queries.*
7. *Drag the merged object [Company No.] and [Company Currency] to create two levels of sections.*
8. *Refresh the report.*

■

5.5 Data Foundation

This section describes central aspects of the data foundation for the Job Budgeting universe.

5.5.1 Fact Tables

The universe contains four fact tables for representing job budgets on three different levels and for representing purchase orders associated job budget lines. The tables are described in detail in the following sections.

JOBBUDGETLINE_S

This fact table represents job budget lines. The fact table is defined as an alias of the dimension table `JOBBUDGETLINE_D` which basically selects fields from the EX view applied to the database table `JOBBUDGETLINE`. The derived table, however, filters off certain lines. Job budget lines can in Maconomy be organized to form a hierarchy, which means that some lines are *parent* lines and some are leaves. From a reporting point of view, we are only interested in the leaves; i.e. the information-bearing lines and not the `"sum/text"` lines. Therefore, we filter off the job budget lines having sub-lines.

Furthermore, we filter off the job budget lines that are defined as summation lines using the old-fashioned way. That way uses a blank activity number or an activity number equal to a star (∗).

JOBBUDGETJOBBUDGETREVISION_D

The header table of the `JOBBUDGETLINE` table is the `JOBBUDGET` table. However, if the job budget has been approved it becomes a *revision* and then a record is also added to the table `JOBBUDGETREVISION`. This means that `JOBBUDGETLINE` actually has two header tables:

- `JOBBUDGET` which represents the latest job budget revision. The job budget lines that are associated with it, represent the lines of the latest revision.

- JOBBUDGETREVISION which represents the approved revisions. The job budget lines that are associated with it, represent the lines of approved job budget revisions.

The two header tables actually store several dimension and measure fields in summarized form. Therefore, we utilize them in aggregate awareness. This means that if the budget figures are only combined with dimension objects about company and job, we can use the aggregated tables instead of JOBBUDGETLINE. However, as soon as we bring in dimensions like task, activity or standard dimensions, the aggregate awareness will choose to use JOBBUDGETLINE (i.e. our JOBBUDGETLINE_S fact table).

Because the two header tables represent different states of a job budget, we combine them in a union to form a comprised header table. The first select statement fetches from the JOBBUDGET table. The second select statement fetches from the JOBBUDGETREVISION table.

R The two header tables JOBBUDGET and JOBBUDGETREVISION actually *overlap*. The former represents the latest revision. The latter represents the approved revisions. Thus, if the latest revision is approved, one of the header records are superfluous. This is handled in the table JOBBUDGETJOBBUDGETREVISION_D by inner joining the JOBBUDGETREVISION to the JOBBUDGET and then filter off the records from JOBBUDGETREVISION which have the same revision number as the JOBBUDGET record.

EXJOBBUDGETLINEPERIOD

This fact table is for providing the ability to report on periodic job budgets. In Maconomy, periodic job budgets are structured over JOBBUDGETLINE records, and for each record there may be one or more records in the table JOBBUDGETLINEPERIOD. Each record represents a value for a certain period and job budget line. The table is *sparse* in the sense that only values different from zero, are stored.

We apply the EX view on JOBBUDGETLINEPERIOD to get real date fields and informative popups.

In the data foundation, the fact table is joined to the dimension table JOBBUDGETLINE_D from where all the line specific dimension fields are fetched.

R The fact tables JOBBUDGETJOBBUDGETREVISION and EXJOBBUDGETLINEPERIOD do not themselves apply the BO view for access control. For performance reasons, this is done implicitly by forcing the objects defined on the tables, to include the dimension table JOBHEADER_D. Thereby, access is applied through the job which is the access definition of the tables anyway.

BOPURCHASEORDERLINE

Job budget lines can have purchases associated. The fact table BOPURCHASEORDERLINE is joined to the dimension table JOBBUDGETLINE_D to provide the ability to report on such purchases.

Chapter 6

Job Information Universe

The Job Information universe provides objects for reporting on the setup and status of jobs. Besides the ability to report on various dimensions and characteristics of jobs (like whether it is invoiceable or not), the universe contains objects for reporting on the following job related aspects:

- Job group information
- Main-job Sub-job relationships
- Status of jobs (e.g. whether closed, blocked for budgeting, etc.)
- Job Phases
- Origins (whether created from template job or opportunity)
- Current budget types
- Approval status and proposal abilities
- Invoice status (e.g. lastest invoice number and the number of current drafts)
- Employee control
- Price lists
- Pending job actions
- Task lists, default tasks, task groups, overwriting principles of tasks lists
- Customer Tax Return
- Job parameters, attributes and job parameter selections
- Job collections

Furthermore, it is possible to report on dimension information like the delivery and payment customers of the job, bill to customer distribution, the company and employees associated the job (including project manager and account manager)

The universe does not provide any measure objects; only dimensions. It can be used on its own for reporting on job setup or in combination with queries to other universes like the Job Invoicing universe. Thereby, complex reports can be written which take the measures from the Job Invoicing universe and advanced job setup data from the Job Information universe.

6.1 Prerequisites

There are no prerequisites for using this universe.

Chapter 7

Time Sheet Universe

The Time Sheet universe provides objects that allow you to report on time sheet registrations and the status of time sheets: *registered*, *submitted*, *approved* (by superior, project manager or both), *released*, etc. It is possible to report both on weekly summations and individual days. The universe will automatically figure out whether to take figures from the weekly time sheet or the daily time sheet, depending on the dimensions the measures are combined or sliced by. It is also possible to report on the hourly rate (unit price) that is valid at the time and stage of the time sheet registration. Combined with e.g. the unsubmitted hours, it is possible to get an estimation of the potential revenue that the unsubmitted hours represent.

The measure objects can be grouped and restricted by a variety of dimensions including company, customer, job, activity, task, employees, dates, and the ten standard dimensions.

7.1 Prerequisites

In order to fully benefit from the Time Sheet universe, Maconomy should be set up such that:

- The system parameter **Use Calendar Day** is enabled and the calendar spans all necessary dates.
- Week calendars must be set up for all employees.
- Fixed working hours must be set up for all employees.
- The system parameter **Use Daily Time Sheets** must be enabled.

7.2 Maconomy Workflows

Time registration can be done in several ways and several windows in Maconomy. In the following we shall distinguish between weekly and daily time registrations.

The weekly time registrations are typically done in the **Time Sheets** window. Several windows can be used for approval; including the **Time Sheets** window itself. A weekly time sheet is represented by a record in the table `TIMESHEETHEADER` and a collection of records in the table `TIMESHEETLINE`. There can only be one weekly time sheet for each combination of employee an year/week.

Daily time sheet registrations are typically done in the **SpeedSheet** window or the **Daily Time Sheets** window. Daily time sheets are represented by a record in the table `DAILYTIMESHEETHEADER` and a collection of records in the table `DAILYTIMESHEETLINE`. There can only be one daily time sheet for each combination of employee and date.

Daily time sheets are both associated weekly time sheet concerning data and workflow. The daily registrations are reflected on the weekly time sheets, and when all five work days are submitted, the corresponding weekly time sheet is also submitted.

7.2.1 Weekly Time Sheet Registrations

Consider the following scenario:

1. In the window **Jobs** create a new job.
2. Set up the job for both requiring approval by line and project manager.
3. In the window **Time Sheets** create a new time sheet for the current week and a chosen employee.
4. In the lower panel of the window, create a line, state the job number, and enter an amount of 5 hours for each of the five working days.

   This will provide a record in the table `TIMESHEETHEADER` and a record in the table `TIMESHEETLINE`. The time sheet header will have status of not submitted *and* not approved. *The `LINEAPPROVALSTATUS` field states whether the project manager needs to approve the individual lines. At this stage, nothing is yet approved as the time sheet is not submitted yet.*
5. In the window **Time Sheet Approval** approve as project manager.

   This will change the status of the time sheet line to approved and the field `LINEAPPROVALSTATUS` will state approved. However, as the job also requires line manager approval, the status field `FULLYAPPROVED` will not state that the time sheet is approved yet.
6. In the window **Time Sheets** approve the time sheet as line manager of the employee.

The status field `FULLYAPPROVED` *will now state that the time sheet is fully approved. If the time sheet have had more lines, each of these would have to be approved by the project manager until the time sheet would be fully approved.*

7.2.2 Daily Time Sheet Registrations

Consider a similar scenario just using **SpeedSheet**:

1. In the window **SpeedSheet** create a line for the current date and the chosen employee.
2. State the job and an amount of 5 hours for that date.

   This will provide a record in the `DAILYTIMESHEETHEADER` *and a record in the* `DAILYTIMESHEETLINE` *tables. Correspondingly, records will also be provided in the* `TIMESHEETHEADER` *and* `TIMESHEETLINE` *tables.*

The approval principles are the same for daily and weekly time sheets. In the previous scenario, we have seen that daily time sheet registrations reflects on weekly time sheets. In fact the other way around is also the case. When registering in a weekly time sheet the corresponding daily time sheets are automatically created or updated.

R The use of daily time sheets assume that this functionality is enabled in the system parameters. In BPM, it is an assumption that this is the case.

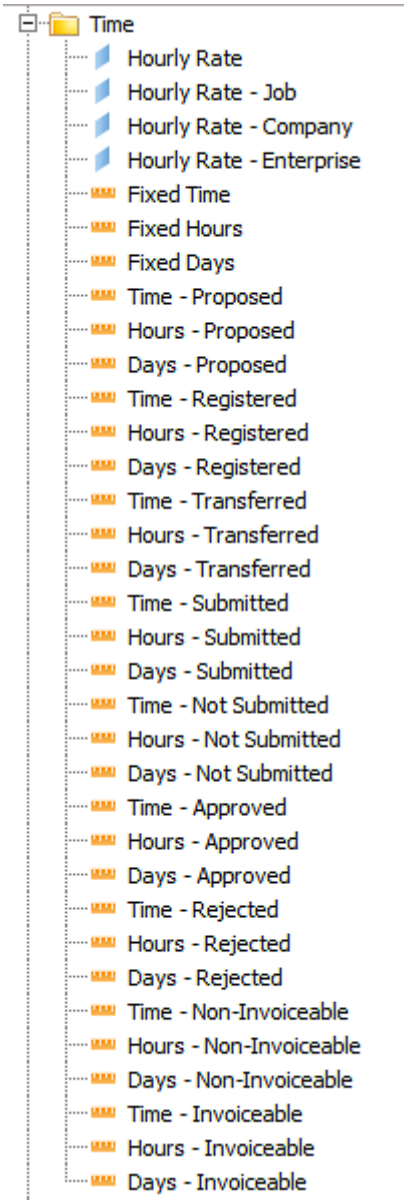
7.3 Business Layer

The universe provides measures for reporting on time sheet registrations and dimensions for reporting and filtering on associated values like the jobs, activities and tasks entered on the registrations.

The objects for reporting on registered time are located in the [`Hours`] folder. There are objects for displaying the figures working hours, and objects for displaying the hours on the various submit-approval workflow stages.

In the following table—for simplicity—we assume the aggregation level of `TIMESHEETLINE` but similar expressions are used for the level of `DAILYTIMESHEETLINE`. Some objects are even defined as well on the `TIMESHEETHEADER` table which boosts the performance in general.

Fixed hours are calculated in the same way as done in the Utilization universe (see Section 8.3.5.) The objects for the individual submit-approval stages use properties of the time sheet as well as approval properties set up on the job. The objects for registered hours, days or time simply state the values that the user has entered on the time sheet. The submitted values are the registered for which the status of the time sheet header is submitted. For approved values, we additionally need to check the principles of line manager approval and project manager approval, as set up on the job. We only consider hours, days or time as approved if both the line manager and the project manager have approved; i.e. if they are required to approve. If only line manager approval is required, project manager approval is ignored, etc.



[Fixed Time]

$$max \begin{cases} \texttt{DAYCALENDARLINE.FIXEDWORKINGTIME} \\ FixedEmployeeTime \end{cases}$$

where $FixedEmployeeTime =$

EMPLOYEEREVISION.FIXEDWORKINGTIMEMONDAY
  if CALENDARDAY.DAYOFWEEK $= 1$
EMPLOYEEREVISION.FIXEDWORKINGTIMETUESDAY
  if CALENDARDAY.DAYOFWEEK $= 2$
EMPLOYEEREVISION.FIXEDWORKINGTIMEWEDNESDAT
  if CALENDARDAY.DAYOFWEEK $= 3$
EMPLOYEEREVISION.FIXEDWORKINGTIMETHURSDAY
  if CALENDARDAY.DAYOFWEEK $= 4$
EMPLOYEEREVISION.FIXEDWORKINGTIMEFRIDAY
  if CALENDARDAY.DAYOFWEEK $= 5$
EMPLOYEEREVISION.FIXEDWORKINGTIMESATURDAY
  if CALENDARDAY.DAYOFWEEK $= 6$
EMPLOYEEREVISION.FIXEDWORKINGTIMESUNDAY
  if CALENDARDAY.DAYOFWEEK $= 7$

[Fixed Hours]

$$max \begin{cases} \texttt{DAYCALENDARLINE.FIXEDWORKINGTIME} \\ FixedEmployeeTime \end{cases}$$
$$if\ \texttt{EXEMPLOYEEREVISION.TIMEREGISTRATIONUNITPN} = 0$$

$$\texttt{EXEMPLOYEEREVISION.HOURSPERMANDAY} * max \begin{cases} \texttt{DAYCALENDARLINE.FIXEDWORKINGTIME} \\ FixedEmployeeTime \end{cases}$$
$$if\ \texttt{EXEMPLOYEEREVISION.TIMEREGISTRATIONUNITPN} = 1$$

$$where\ FixedEmployeeTime =$$
$$\texttt{EMPLOYEEREVISION.FIXEDWORKINGTIMEMONDAY}\ if\ \texttt{CALENDARDAY.DAYOFWEEK} = 1$$
$$\texttt{EMPLOYEEREVISION.FIXEDWORKINGTIMETUESDAY}\ if\ \texttt{CALENDARDAY.DAYOFWEEK} = 2$$
$$\texttt{EMPLOYEEREVISION.FIXEDWORKINGTIMEWEDNESDAT}\ if\ \texttt{CALENDARDAY.DAYOFWEEK} = 3$$
$$\texttt{EMPLOYEEREVISION.FIXEDWORKINGTIMETHURSDAY}\ if\ \texttt{CALENDARDAY.DAYOFWEEK} = 4$$
$$\texttt{EMPLOYEEREVISION.FIXEDWORKINGTIMEFRIDAY}\ if\ \texttt{CALENDARDAY.DAYOFWEEK} = 5$$
$$\texttt{EMPLOYEEREVISION.FIXEDWORKINGTIMESATURDAY}\ if\ \texttt{CALENDARDAY.DAYOFWEEK} = 6$$
$$\texttt{EMPLOYEEREVISION.FIXEDWORKINGTIMESUNDAY}\ if\ \texttt{CALENDARDAY.DAYOFWEEK} = 7$$

[Fixed Days]

$$max \begin{cases} \texttt{DAYCALENDARLINE.FIXEDWORKINGTIME} \\ FixedEmployeeTime \end{cases}$$
if EXEMPLOYEEREVISION.TIMEREGISTRATIONUNITPN $= 1$

$$\frac{max \begin{cases} \texttt{DAYCALENDARLINE.FIXEDWORKINGTIME} \\ FixedEmployeeTime \end{cases}}{\texttt{EXEMPLOYEEREVISION.HOURSPERMANDAY}}$$

if EXEMPLOYEEREVISION.TIMEREGISTRATIONUNITPN $= 0$
and EXEMPLOYEEREVISION.HOURSPERMANDAY > 0.001

$0 otherwise$

where $FixedEmployeeTime =$
EMPLOYEEREVISION.FIXEDWORKINGTIMEMONDAY
if CALENDARDAY.DAYOFWEEK $= 1$
EMPLOYEEREVISION.FIXEDWORKINGTIMETUESDAY
if CALENDARDAY.DAYOFWEEK $= 2$
EMPLOYEEREVISION.FIXEDWORKINGTIMEWEDNESDAT
if CALENDARDAY.DAYOFWEEK $= 3$
EMPLOYEEREVISION.FIXEDWORKINGTIMETHURSDAY
if CALENDARDAY.DAYOFWEEK $= 4$
EMPLOYEEREVISION.FIXEDWORKINGTIMEFRIDAY
if CALENDARDAY.DAYOFWEEK $= 5$
EMPLOYEEREVISION.FIXEDWORKINGTIMESATURDAY
if CALENDARDAY.DAYOFWEEK $= 6$
EMPLOYEEREVISION.FIXEDWORKINGTIMESUNDAY
if CALENDARDAY.DAYOFWEEK $= 7$

[Fixed Days]

$$max \begin{cases} \texttt{DAYCALENDARLINE.FIXEDWORKINGTIME} \\ FixedEmployeeTime \end{cases}$$
if EXEMPLOYEEREVISION.TIMEREGISTRATIONUNITPN $= 1$

$$\frac{max \begin{cases} \texttt{DAYCALENDARLINE.FIXEDWORKINGTIME} \\ FixedEmployeeTime \end{cases}}{\texttt{EXEMPLOYEEREVISION.HOURSPERMANDAY}}$$

if EXEMPLOYEEREVISION.TIMEREGISTRATIONUNITPN $= 0$
and EXEMPLOYEEREVISION.HOURSPERMANDAY > 0.001

0 *otherwise*

where $FixedEmployeeTime =$
EMPLOYEEREVISION.FIXEDWORKINGTIMEMONDAY
if CALENDARDAY.DAYOFWEEK $= 1$
EMPLOYEEREVISION.FIXEDWORKINGTIMETUESDAY
if CALENDARDAY.DAYOFWEEK $= 2$
EMPLOYEEREVISION.FIXEDWORKINGTIMEWEDNESDAT
if CALENDARDAY.DAYOFWEEK $= 3$
EMPLOYEEREVISION.FIXEDWORKINGTIMETHURSDAY
if CALENDARDAY.DAYOFWEEK $= 4$
EMPLOYEEREVISION.FIXEDWORKINGTIMEFRIDAY
if CALENDARDAY.DAYOFWEEK $= 5$
EMPLOYEEREVISION.FIXEDWORKINGTIMESATURDAY
if CALENDARDAY.DAYOFWEEK $= 6$
EMPLOYEEREVISION.FIXEDWORKINGTIMESUNDAY
if CALENDARDAY.DAYOFWEEK $= 7$

[Time - Proposed]
TIMESHEETLINE.NUMBERPROPOSED

[Hours - Proposed]
TIMESHEETLINE.NUMBERPROPOSED
if EXEMPLOYEEREVISION.TIMEREGISTRATIONUNITPN $= 0$

EXEMPLOYEEREVISION.HOURSPERMANDAY
$*$TIMESHEETLINE.NUMBERPROPOSED
otherwise

[Days - Proposed]

$$\begin{cases} \texttt{TIMESHEETLINE.NUMBERPROPOSED} & \textit{if } \texttt{EXEMPLOYEEREVISION.TIMEREGISTRATIONUNITPN} = 1 \\ \dfrac{\texttt{TIMESHEETLINE.NUMBERPROPOSED}}{\texttt{EXEMPLOYEEREVISION.HOURSPERMANDAY}} & \textit{if } \texttt{EXEMPLOYEEREVISION.TIMEREGISTRATIONUNITPN} = 0 \\ & \textit{and } \texttt{EXEMPLOYEEREVISION.HOURSPERMANDAY} > 0.001 \\ \textit{otherwise} & \end{cases}$$

[Time - Registered]

TIMESHEETLINE.WEEKTOTAL

[Hours - Registered]

$$\begin{cases} \texttt{TIMESHEETLINE.WEEKTOTAL} & \textit{if } \texttt{EXEMPLOYEEREVISION.TIMEREGISTRATIONUNITPN} = 0 \\ \texttt{EXEMPLOYEEREVISION.HOURSPERMANDAY} & \\ *\texttt{TIMESHEETLINE.NUMBERPROPOSED} & \\ \textit{otherwise} & \end{cases}$$

[Days - Registered]

$$\begin{cases} \texttt{TIMESHEETLINE.WEEKTOTAL} & \textit{if } \texttt{EXEMPLOYEEREVISION.TIMEREGISTRATIONUNITPN} = 1 \\ \dfrac{\texttt{TIMESHEETLINE.NUMBERPROPOSED}}{\texttt{EXEMPLOYEEREVISION.HOURSPERMANDAY}} & \textit{if } \texttt{EXEMPLOYEEREVISION.TIMEREGISTRATIONUNITPN} = 0 \\ & \textit{and } \texttt{EXEMPLOYEEREVISION.HOURSPERMANDAY} > 0.001 \\ 0 \textit{ otherwise} & \end{cases}$$

[Time - Transferred]

$$\begin{cases} \texttt{TIMESHEETLINE.NUMBERDAY1TRANSFERRED} \\ + \texttt{TIMESHEETLINE.NUMBERDAY2TRANSFERRED} \\ \vdots \\ + \texttt{TIMESHEETLINE.NUMBERDAY7TRANSFERRED} \end{cases}$$

[Hours - Transferred]

$$\begin{aligned}
&\texttt{TIMESHEETLINE.NUMBERDAY1TRANSFERRED} \\
&+ \texttt{TIMESHEETLINE.NUMBERDAY2TRANSFERRED} \\
&\vdots \\
&+ \texttt{TIMESHEETLINE.NUMBERDAY7TRANSFERED} \\
&\textit{if } \texttt{EXEMPLOYEEREVISION.TIMEREGISTRATIONUNITPN} = 0 \\
&\\
&\texttt{EXEMPLOYEEREVISION.HOURSPERMANDAY} \\
&*(\texttt{TIMESHEETLINE.NUMBERDAY1TRANSFERRED} \\
&\quad + \texttt{TIMESHEETLINE.NUMBERDAY2TRANSFERRED} \\
&\quad \vdots \\
&\quad + \texttt{TIMESHEETLINE.NUMBERDAY7TRANSFERED}) \\
&\textit{if } \texttt{EXEMPLOYEEREVISION.TIMEREGISTRATIONUNITPN} = 1 \\
&\\
&0 \textit{ otherwise}
\end{aligned}$$

[Hours - Transferred]

$$\begin{aligned}
&\texttt{TIMESHEETLINE.NUMBERDAY1TRANSFERRED} \\
&+ \texttt{TIMESHEETLINE.NUMBERDAY2TRANSFERRED} \\
&\vdots \\
&+ \texttt{TIMESHEETLINE.NUMBERDAY7TRANSFERED} \\
&\textit{if } \texttt{EXEMPLOYEEREVISION.TIMEREGISTRATIONUNITPN} = 1 \\
&\\
&\frac{1}{\texttt{EXEMPLOYEEREVISION.HOURSPERMANDAY}} \\
&*(\texttt{TIMESHEETLINE.NUMBERDAY1TRANSFERRED} \\
&\quad + \texttt{TIMESHEETLINE.NUMBERDAY2TRANSFERRED} \\
&\quad \vdots \\
&\quad + \texttt{TIMESHEETLINE.NUMBERDAY7TRANSFERED}) \\
&\textit{if } \texttt{EXEMPLOYEEREVISION.TIMEREGISTRATIONUNITPN} = 0 \\
&\quad \textit{and } \texttt{EXEMPLOYEEREVISION.HOURSPERMANDAY} > 0.001 \\
&\\
&0 \textit{ otherwise}
\end{aligned}$$

[Time - Submitted]

$$\begin{aligned}
&\texttt{TIMESHEETLINE.WEEKTOTAL} \\
&\textit{if } \texttt{EXTIMESHEETHEADER.SUBMITTEDPN} = 1
\end{aligned}$$

[Hours - Submitted]

TIMESHEETLINE.WEEKTOTAL
if EXTIMESHEETHEADER.SUBMITTEDPN = 1
and EXEMPLOYEEREVISION.TIMEREGISTRATIONPN = 0

EXEMPLOYEEREVISION.HOURSPERMANDAY * TIMESHEETLINE.WEEKTOTAL
if EXTIMESHEETHEADER.SUBMITTEDPN = 1
and EXEMPLOYEEREVISION.TIMEREGISTRATIONPN = 1

0 *otherwise*

[Days - Submitted]

TIMESHEETLINE.WEEKTOTAL
if EXTIMESHEETHEADER.SUBMITTEDPN = 1
and EXEMPLOYEEREVISION.TIMEREGISTRATIONPN = 1

$$\frac{\texttt{TIMESHEETLINE.WEEKTOTAL}}{\texttt{EXEMPLOYEEREVISION.HOURSPERMANDAY}}$$

if EXTIMESHEETHEADER.SUBMITTEDPN = 1
and EXEMPLOYEEREVISION.TIMEREGISTRATIONPN = 0
and EXEMPLOYEEREVISION.HOURSPERMANDAY > 0.001

0 *otherwise*

[Time - Not Submitted]

TIMESHEETLINE.WEEKTOTAL
if EXTIMESHEETHEADER.SUBMITTEDPN = 0

[Hours - Not Submitted]

TIMESHEETLINE.WEEKTOTAL
if EXTIMESHEETHEADER.SUBMITTEDPN = 0
and EXEMPLOYEEREVISION.TIMEREGISTRATIONPN = 0

EXEMPLOYEEREVISION.HOURSPERMANDAY * TIMESHEETLINE.WEEKTOTAL
if EXTIMESHEETHEADER.SUBMITTEDPN = 0
and EXEMPLOYEEREVISION.TIMEREGISTRATIONPN = 1

0 *otherwise*

[Days - Not Submitted]

$$\begin{cases} \texttt{TIMESHEETLINE.WEEKTOTAL} \\ \textit{if } \texttt{EXTIMESHEETHEADER.SUBMITTEDPN} = 0 \\ \quad \textit{and } \texttt{EXEMPLOYEEREVISION.TIMEREGISTRATIONPN} = 1 \\ \\ \dfrac{\texttt{TIMESHEETLINE.WEEKTOTAL}}{\texttt{EXEMPLOYEEREVISION.HOURSPERMANDAY}} \\ \textit{if } \texttt{EXTIMESHEETHEADER.SUBMITTEDPN} = 0 \\ \quad \textit{and } \texttt{EXEMPLOYEEREVISION.TIMEREGISTRATIONPN} = 0 \\ \quad \textit{and } \texttt{EXEMPLOYEEREVISION.HOURSPERMANDAY} > 0.001 \\ \\ 0 \textit{ otherwise} \end{cases}$$

[Time - Approved]

$$\begin{cases} \texttt{TIMESHEETLINE.WEEKTOTAL} \\ \textit{if } \texttt{EXTIMESHEETHEADER.SUBMITTEDPN} = 1 \\ \textit{and } (\texttt{EXJOBHEADER.APPROVALSUPERIORPN} \neq 0 \\ \quad \textit{or } \texttt{EXTIMESHEETHEADER.APPROVEDPN} = 1) \\ \textit{and } (\texttt{EXJOBHEADER.APPROVALPROJECTMANAGERPN} \neq 0 \\ \quad \textit{or } \texttt{EXTIMESHEETHEADER.LINEAPPROVALSTATUSPN} = 0) \\ \\ 0 \textit{ otherwise} \end{cases}$$

[Hours - Approved]

$$\begin{cases} \texttt{TIMESHEETLINE.WEEKTOTAL} \\ \textit{if } \texttt{EXTIMESHEETHEADER.SUBMITTEDPN} = 1 \\ \textit{and } \texttt{EXEMPLOYEEREVISION.TIMEREGISTRATIONPN} = 0 \\ \textit{and } (\texttt{EXJOBHEADER.APPROVALSUPERIORPN} \neq 0 \\ \quad \textit{or } \texttt{EXTIMESHEETHEADER.APPROVEDPN} = 1) \\ \textit{and } (\texttt{EXJOBHEADER.APPROVALPROJECTMANAGERPN} \neq 0 \\ \quad \textit{or } \texttt{EXTIMESHEETHEADER.LINEAPPROVALSTATUSPN} = 0) \\ \\ \texttt{TIMESHEETLINE.WEEKTOTAL} * \texttt{EXEMPLOYEEREVISION.HOURSPERMANDAY} \\ \textit{if } \texttt{EXTIMESHEETHEADER.SUBMITTEDPN} = 1 \\ \textit{and } \texttt{EXEMPLOYEEREVISION.TIMEREGISTRATIONPN} = 1 \\ \textit{and } (\texttt{EXJOBHEADER.APPROVALSUPERIORPN} \neq 0 \\ \quad \textit{or } \texttt{EXTIMESHEETHEADER.APPROVEDPN} = 1) \\ \textit{and } (\texttt{EXJOBHEADER.APPROVALPROJECTMANAGERPN} \neq 0 \\ \quad \textit{or } \texttt{EXTIMESHEETHEADER.LINEAPPROVALSTATUSPN} = 0) \\ \\ 0 \textit{ otherwise} \end{cases}$$

[Days - Approved]

TIMESHEETLINE.WEEKTOTAL
if EXTIMESHEETHEADER.SUBMITTEDPN = 1
and EXEMPLOYEEREVISION.TIMEREGISTRATIONPN = 1
and (EXJOBHEADER.APPROVALSUPERIORPN ≠ 0
or EXTIMESHEETHEADER.APPROVEDPN = 1)
and (EXJOBHEADER.APPROVALPROJECTMANAGERPN ≠ 0
or EXTIMESHEETHEADER.LINEAPPROVALSTATUSPN = 0)

$$\frac{\texttt{TIMESHEETLINE.WEEKTOTAL}}{\texttt{EXEMPLOYEEREVISION.HOURSPERMANDAY}}$$

if EXTIMESHEETHEADER.SUBMITTEDPN = 1
and EXEMPLOYEEREVISION.TIMEREGISTRATIONPN = 0
and EXEMPLOYEEREVISION.HOURSPERMANDAY > 0.001
and (EXJOBHEADER.APPROVALSUPERIORPN ≠ 0
or EXTIMESHEETHEADER.APPROVEDPN = 1)
and (EXJOBHEADER.APPROVALPROJECTMANAGERPN ≠ 0
or EXTIMESHEETHEADER.LINEAPPROVALSTATUSPN = 0)

0 *otherwise*

[Time - Rejected]

TIMESHEETLINE.WEEKTOTAL
if EXTIMESHEETHEADER.LINEAPPROVALSTATUSPN = 1

0 *otherwise*

[Hours - Rejected]

TIMESHEETLINE.WEEKTOTAL
if EXTIMESHEETHEADER.LINEAPPROVALSTATUSPN = 1
and EXEMPLOYEEREVISION.TIMEREGISTRATIONPN = 0

TIMESHEETLINE.WEEKTOTAL * EXEMPLOYEEREVISION.HOURSPERMANDAY
if EXTIMESHEETHEADER.LINEAPPROVALSTATUSPN = 1
and EXEMPLOYEEREVISION.TIMEREGISTRATIONPN = 1

0 *otherwise*

[Days - Rejected]

$$\begin{cases} \texttt{TIMESHEETLINE.WEEKTOTAL} \\ \textit{if } \texttt{EXTIMESHEETHEADER.LINEAPPROVALSTATUSPN} = 1 \\ \textit{and } \texttt{EXEMPLOYEEREVISION.TIMEREGISTRATIONPN} = 1 \\ \\ \dfrac{\texttt{TIMESHEETLINE.WEEKTOTAL}}{\texttt{EXEMPLOYEEREVISION.HOURSPERMANDAY}} \\ \textit{if } \texttt{EXTIMESHEETHEADER.LINEAPPROVALSTATUSPN} = 1 \\ \textit{and } \texttt{EXEMPLOYEEREVISION.HOURSPERMANDAY} > 0.001 \\ \textit{and } \texttt{EXEMPLOYEEREVISION.TIMEREGISTRATIONPN} = 0 \\ \\ 0 \ \textit{otherwise} \end{cases}$$

[Time - Non-Invoiceable]

$$\begin{cases} \texttt{TIMESHEETLINE.NUMBEROF} \\ \textit{if } \texttt{EXACTIVITY.TOBEINVOICEDPN} = 0 \\ \quad \textit{or } \texttt{EXJOBHEADER.NOTINVOICEABLEPN} = 1 \\ \\ 0 \ \textit{otherwise} \end{cases}$$

[Hours - Non-Invoiceable]

$$\begin{cases} \texttt{TIMESHEETLINE.NUMBEROF} \\ \textit{if } \texttt{EXEMPLOYEEREVISION.TIMEREGISTRATIONPN} = 0 \\ \textit{and } (\texttt{EXACTIVITY.TOBEINVOICEDPN} = 0 \\ \quad \textit{or } \texttt{EXJOBHEADER.NOTINVOICEABLEPN} = 1) \\ \\ \texttt{TIMESHEETLINE.NUMBEROF} * \texttt{EXEMPLOYEEREVISION.HOURSPERMANDAY} \\ \textit{if } \texttt{EXEMPLOYEEREVISION.TIMEREGISTRATIONPN} = 1 \\ \textit{and } (\texttt{EXACTIVITY.TOBEINVOICEDPN} = 0 \\ \quad \textit{or } \texttt{EXJOBHEADER.NOTINVOICEABLEPN} = 1) \\ \\ 0 \ \textit{otherwise} \end{cases}$$

[Days - Non-Invoiceable]

TIMESHEETLINE.NUMBEROF
if EXEMPLOYEEREVISION.TIMEREGISTRATIONPN = 1
and (EXACTIVITY.TOBEINVOICEDPN = 0
or EXJOBHEADER.NOTINVOICEABLEPN = 1)

$$\frac{\texttt{TIMESHEETLINE.NUMBEROF}}{\texttt{EXEMPLOYEEREVISION.HOURSPERMANDAY}}$$

if EXEMPLOYEEREVISION.TIMEREGISTRATIONPN = 0
and EXEMPLOYEEREVISION.HOURSPERMANDAY > 0.001
and (EXACTIVITY.TOBEINVOICEDPN = 0
or EXJOBHEADER.NOTINVOICEABLEPN = 1)

0 *otherwise*

[Time - Invoiceable]

TIMESHEETLINE.NUMBEROF
if EXACTIVITY.TOBEINVOICEDPN = 1
and EXJOBHEADER.NOTINVOICEABLEPN = 0

0 *otherwise*

[Hours - Invoiceable]

TIMESHEETLINE.NUMBEROF
if EXEMPLOYEEREVISION.TIMEREGISTRATIONPN = 0
and EXACTIVITY.TOBEINVOICEDPN = 1
and EXJOBHEADER.NOTINVOICEABLEPN = 0

TIMESHEETLINE.NUMBEROF * EXEMPLOYEEREVISION.HOURSPERMANDAY
if EXEMPLOYEEREVISION.TIMEREGISTRATIONPN = 1
and EXACTIVITY.TOBEINVOICEDPN = 1
and EXJOBHEADER.NOTINVOICEABLEPN = 0

0 *otherwise*

[Days - Invoiceable]

$$
\begin{cases}
\texttt{TIMESHEETLINE.NUMBEROF} \\
\quad \textit{if } \texttt{EXEMPLOYEEREVISION.TIMEREGISTRATIONPN} = 1 \\
\quad \textit{and } \texttt{EXACTIVITY.TOBEINVOICEDPN} = 1 \\
\quad\quad \textit{and } \texttt{EXJOBHEADER.NOTINVOICEABLEPN} = 0 \\
\\
\dfrac{\texttt{TIMESHEETLINE.NUMBEROF}}{\texttt{EXEMPLOYEEREVISION.HOURSPERMANDAY}} \\
\quad \textit{if } \texttt{EXEMPLOYEEREVISION.TIMEREGISTRATIONPN} = 0 \\
\quad \textit{and } \texttt{EXEMPLOYEEREVISION.HOURSPERMANDAY} > 0.001 \\
\quad \textit{and } \texttt{EXACTIVITY.TOBEINVOICEDPN} = 1 \\
\quad\quad \textit{and } \texttt{EXJOBHEADER.NOTINVOICEABLEPN} = 0 \\
\\
0 \ \textit{otherwise}
\end{cases}
$$

Chapter 8

Utilization Universe

The Utilization universe provides objects for reporting on the utilization of employees. It provides the ability to report on registered hours, billing prices, and comparing these to the fixed hours defined for the employee or in the week calendar used by the employee. It is possible to distinguish the different kinds of registrations like invoiceable and non-invoiceable hours. It is also possible to utilize the employee utilization popup on activities for categorizing registrations. The universe includes a set of suggested utilization metrics which pre-calculate different utilization figures based on the PSO solutions setup of the employee utilization popups on activities.

The access control in the Utilization universe goes through the employee being reported on and not the access to registrations and jobs. Thereby, it differs from the access control approach in the other universes. The reason for the approach is that it makes it possible for department managers to report on their employee's utilization even though some employees may have registered time on external jobs that the department manager does not have access to.

The measure of the universe, can be grouped and restricted by a variety of dimensions including company, customer, job, employee category, employee, activity, task, dates, and the ten standard dimensions.

8.1 Prerequisites

In order to fully benefit from the Utilization universe, Maconomy should be set up such that:

- The system parameter **Use Calendar Day** is enabled and the calendar spans all necessary dates.
- The **Employee Utilization** popup should be setup to have the following values (0-9):

  Productive Hours

Invoiceable

Productive hours

Non-Invoiceable

Non-Productive

Absence

New Business/Sales

Training

Sickness

Holiday

- Week calendars are setup and assigned for all employees.
- Fixed hours are setup and maintained in the employee revisions of each employee.
- Dimensions of employees are setup and maintained in the derived dimensions.

8.2 Maconomy Workflows

The Utilization universe is different from any other universe in BPM in the way access control is defined. Normally, access control is defined on each table so that access matches access control in Maconomy; as defined by standard. However, in the Utilization universe, access is defined through the employee. This means that if the user has access to the employee, the user has access to registrations of that employee. This is necessary because utilization reports are often made for line or department managers. These employees have access to the employees they manage, but not necessarily the jobs the employee's registered time on. Normal access control would disregard registrations on jobs that the manager does not have access to. With access through the employee, such registrations can be included.

8.2.1 Invoiceable and Non-Invoiceable Time

An important distinction in utilization reports, is the distinction between invoiceable and non-invoiceable time. Consider the following scenario:

1. In the **Jobs** create two jobs: an invoiceable job and a non-invoiceable job.
2. For an employee create a new time sheet in the window **Time Sheets**.
3. For *monday* register 5 hours on the invoiceable job using an invoiceable activity, and 2 hours using a non-invoiceable activity.
4. For *tuesday* register 4 hours on the non-invoiceable job using an invoiceable activity, and 1 hour using a non-invoiceable activity.

5. Submit the time sheet.
6. Approve the time sheet.

   The result is typically (but that depends on setup) four job entries:

   - *one with a quantity of* 5 *hours on the invoiceable job and invoiceable activity. These* 5 *hours are invoiceable time.*
   - *one with a quantity of* 2 *hours on the invoiceable job and the non-invoiceable activity. These* 2 *hours are not invoiceable time.*
   - *one with a quantity of* 4 *hours on the non-invoiceable job and the invoiceable activity. These* 4 *hours are not invoiceable time.*
   - *one with a quantity of* 1 *hour on the non-invoiceable job and the non-invoiceable activity. This* 1 *hour is not invoiceable time.*

R Jobs and employees can be setup for registrations in days instead of in hours. In the above scenario, the jobs could have been setup for registration in days which would mean that the quantities we enter are counted in days. E.g. we could have entered a quantity of 0.67 day to achieve the same value as the 5 hours registered in the scenario; assuming that the **number of hours per day** is set to 7.5 for the job.

8.2.2 Employee Revisions and Fixed Time

Fixed working time is defined in two places in Maconomy. The first is in the **Week Calendars** that employees are assigned to use. The second is in the **Employee Revisions** of employees. Each employee revision states a date interval for which it is valid, and it states the fixed employee time, standard dimensions and other dimensions valid for that interval.

When fixed working time is calculated in the Utilization universe, the smallest value is taken of the fixed time defined in the week calendar and the employee revision of the employee. The universe will take care of applying the correct date interval of the revisions and dates of the week calendar, as the dates of the registrations (records in the `JOBENTRY` table) and the fixed time of the week calendar (records in the `WEEKCALENDARLINE` are tied to the date interval of employee revisions.

Consider the following scenario:

1. For an employee open the window **Week Calendars** from the card of the employee.
2. Assign a quantity of 7.5 hours for each working day, except Friday. Assign 5 hours for Friday (lines can be copied.)
3. In the window **Employee Revisions**, assign a quantity of 7 hours for each day (just one revision is needed here.)

This setup provides the foundation for reporting on fixed employee time. The records in table `WEEKCALENDARLINE` *states the fixed time for all employees using that week calendar. The records in the table* `EMPLOYEEREVISION` *states the fixed time for the individual employee. The fixed hours—as calculated by the universe—will be:*

Monday: 7 hours (the employee revision provides the smallest value)

Tuesday: 7 hours (the employee revision provides the smallest value)

Wednesday: 7 hours (the employee revision provides the smallest value)

Thursday: 7 hours (the employee revision provides the smallest value)

Friday: 5 hours (the week calendar provides the smallest value)

8.2.3 Employee Utilization on Activities

Sometimes, making the distinction between invoiceable and non-invoiceable jobs and activities, or the use of tasks and activities, is not sufficient for distinguishing between the kind of registration done by employees. As a dedicated characteristics for utilization reporting, each activity can be assigned an **Employee Utilization** popup literal. The literals are setup in the popup fields for the employee utilization type. The literals can have any desirable string values. We simply choose values that make sense when reporting on utilization.

Consider the following scenario:

1. In the window **Popup Fields**, locate the popups of type *Employee Utilization*.
2. Enter some popup literals if not done already. E.g: Production Time Invoiceable, Production Time Non-Invoiceable, New Business Case, Vacation, etc.
3. In the window **Activities**, locate one of the invoiceable activities used in the previous scenario.
4. Assign one of the popup literals to the field **Employee Utilization**.

   When reporting on registrations using this activity, we can e.g. group by or restrict on the employee utilization. E.g. we can classify registered hours by employee category and display this as a pie chart.

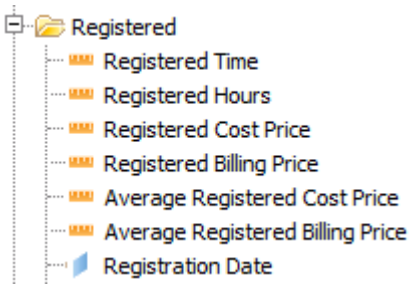
R The names of the popup literals for employee utilization are free-text fields. This means that we can assign any string value as popup literal names. E.g. we can define the name *Production Time, Invoiceable* even though the activity is non-invoiceable. It is entirely up to the person setting up the activities, to choose meaningful literal names.

8.3 Business Layer

The most important measures in utilization reports are the measures for reporting on registered and invoiced time of employees, as well as cost prices and billing prices. Though, the measures in the universe may not comprise a thorough set of all measures possible needed for utilization reporting. The various objects concerning time both come in a *Time* version where time may be in hours or days. This can be determined by the dimension objects [`Employee Time Unit`] and [`Job Time Unit`] that state how it is setup up for employee and job, respectively.

8.3.1 Registered

The objects for reporting on registered figures, are rooted in the `JOBENTRY` table. Thus, registered hours, cost prices and billing prices are basically defined on individual fields on the `JOBENTRY`, similar as done in the Job Invoicing universe. For these objects, we do not distinguish between invoiceable and non-invoiceable figures. Such objects are available in the [`Invoiceable`] and [`Non-Invoiceable`] folders.



[`Registered Time`] `JOBENTRY.NUMBERREGISTERED`

[`Registered Hours`]
`JOBENTRY.NUMBERREGISTERED`
if `EXJOBHEADER.TIMEREGISTRATIONUNITPN` $= 0$
 	
`JOBENTRY.NUMBERREGISTERED` $*$ `JOBHEADER.HOURSPERMANDAY`
otherwise

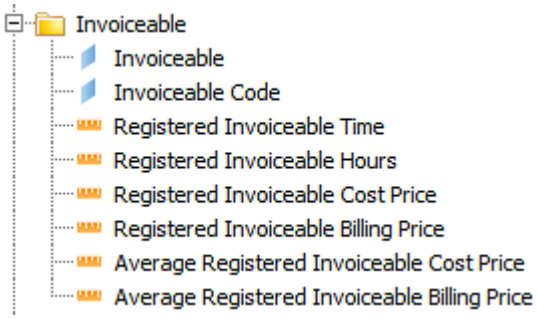
[`Registered Cost Price`] `JOBENTRY.COSTPRICEREG`
[`Registered Billing Price`] `JOBENTRY.BILLINGPRICEREGBASE`
[`Average Registered Cost Price`] $\frac{\texttt{JOBENTRY.COSTPRICEREG}}{\texttt{JOBENTRY.NUMBERREGISTERED}}$ *if* `JOBENTRY.NUMBERREGISTERED` $\neq 0$ 0 *otherwise*

[Average Registered Billing Price]

$$\begin{cases} \dfrac{\texttt{JOBENTRY.BILLINGPRICEREG}}{\texttt{JOBENTRY.NUMBERREGISTERED}} & \textit{if } \texttt{JOBENTRY.NUMBERREGISTERED} \neq 0 \\ 0 & \textit{otherwise} \end{cases}$$

8.3.2 Invoiceable Time

Invoiceable figures are calculated in the same way as the normal registered figures; namely, on fields in the JOBENTRY table (see Section 8.3.1.) The only difference is that for invoiceable figures, only entries where the activity is invoiceable and the job is invoiceable, are included.



[Registered Invoiceable Time]

$$\begin{cases} \texttt{JOBENTRY.NUMBERREGISTERED} & \textit{if } \texttt{EXACTIVITY.TOBEINVOICEDPN} = 1 \\ & \quad \textit{and } \texttt{EXJOBHEADER.NOTINVOICEABLEPN} = 0 \end{cases}$$

[Registered Invoiceable Hours]

$$\begin{cases} \texttt{JOBENTRY.NUMBERREGISTERED} & \textit{if } \texttt{EXJOBHEADER.TIMEREGISTRATIONUNITPN} = 0 \\ & \quad \textit{and } \texttt{EXACTIVITY.TOBEINVOICEDPN} = 1 \\ & \quad \textit{and } \texttt{EXJOBHEADER.NOTINVOICEABLEPN} = 0 \\ \\ \texttt{JOBENTRY.NUMBERREGISTERED} * \texttt{JOBHEADER.HOURSPERMANDAY} & \textit{if } \texttt{EXACTIVITY.TOBEINVOICEDPN} = 1 \\ & \quad \textit{and } \texttt{EXJOBHEADER.NOTINVOICEABLEPN} = 0 \\ \\ 0 & \textit{otherwise} \end{cases}$$

[Registered Invoiceable Cost Price]

$$\begin{cases} \texttt{JOBENTRY.COSTPRICEREG} & \textit{if } \texttt{EXACTIVITY.TOBEINVOICEDPN} = 1 \\ & \quad \textit{and } \texttt{EXJOBHEADER.NOTINVOICEABLEPN} = 0 \end{cases}$$

[Registered Invoiceable Billing Price]

$$\begin{cases} \texttt{JOBENTRY.BILLINGPRICEREGBASE} & \textit{if } \texttt{EXACTIVITY.TOBEINVOICEDPN} = 1 \\ & \quad \textit{and } \texttt{EXJOBHEADER.NOTINVOICEABLEPN} = 0 \end{cases}$$

[Average Registered Invoiceable Cost Price]

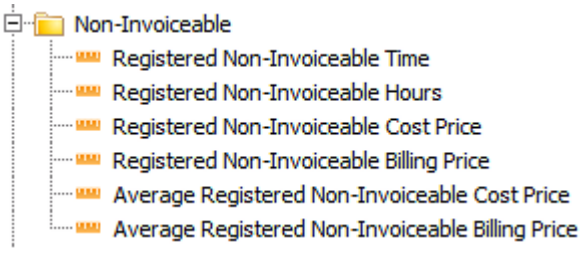
$$\frac{\texttt{JOBENTRY.COSTPRICEREG}}{\texttt{JOBENTRY.NUMBERREGISTERED}}$$
if JOBENTRY.NUMBERREGISTERED $\neq 0$
and EXACTIVITY.TOBEINVOICEDPN $= 1$
and EXJOBHEADER.NOTINVOICEABLEPN $= 0$
0 *otherwise*

[Average Registered Invoiceable Billing Price]

$$\frac{\texttt{JOBENTRY.BILLINGPRICEREG}}{\texttt{JOBENTRY.NUMBERREGISTERED}}$$
if JOBENTRY.NUMBERREGISTERED $\neq 0$
and EXACTIVITY.TOBEINVOICEDPN $= 1$
and EXJOBHEADER.NOTINVOICEABLEPN $= 0$
0 *otherwise*

8.3.3 Non-Invoiceable Time

Non-invoiceable figures are calculated in the same way as the normal registered figures; namely on fields in the JOBENTRY table (see Section 8.3.1) The only difference is that for non-invoiceable figures, only entries where either the activity is non-invoiceable or the job is non-invoiceable, are included.



[Registered Invoiceable Time]

JOBENTRY.NUMBERREGISTERED
if EXACTIVITY.TOBEINVOICEDPN $= 1$
or EXJOBHEADER.NOTINVOICEABLEPN $= 1$

[Registered Invoiceable Hours]

JOBENTRY.NUMBERREGISTERED
if EXJOBHEADER.TIMEREGISTRATIONUNITPN $= 0$
and (EXACTIVITY.TOBEINVOICEDPN $= 0$
or EXJOBHEADER.NOTINVOICEABLEPN $= 1$)

JOBENTRY.NUMBERREGISTERED $*$ JOBHEADER.HOURSPERMANDAY
if EXACTIVITY.TOBEINVOICEDPN $= 1$
and EXJOBHEADER.NOTINVOICEABLEPN $= 0$

0 *otherwise*

[Registered Invoiceable Cost Price] JOBENTRY.COSTPRICEREG *if* EXACTIVITY.TOBEINVOICEDPN = 0 *or* EXJOBHEADER.NOTINVOICEABLEPN = 1
[Registered Invoiceable Billing Price] JOBENTRY.BILLINGPRICEREGBASE *if* EXACTIVITY.TOBEINVOICEDPN = 0 *or* EXJOBHEADER.NOTINVOICEABLEPN = 1
[Average Registered Invoiceable Cost Price] $\frac{\text{JOBENTRY.COSTPRICEREG}}{\text{JOBENTRY.NUMBERREGISTERED}}$ *if* JOBENTRY.NUMBERREGISTERED ≠ 0 *and* (EXACTIVITY.TOBEINVOICEDPN = 0 *or* EXJOBHEADER.NOTINVOICEABLEPN = 1) 0 *otherwise*
[Average Registered Invoiceable Billing Price] $\frac{\text{JOBENTRY.BILLINGPRICEREG}}{\text{JOBENTRY.NUMBERREGISTERED}}$ *if* JOBENTRY.NUMBERREGISTERED ≠ 0 *and* (EXACTIVITY.TOBEINVOICEDPN = 0 *or* EXJOBHEADER.NOTINVOICEABLEPN = 1) 0 *otherwise*

8.3.4 Gross Margins

Gross margins are calculated as the difference between the registered billing price and the registered cost price. There is a version where invoicability is not distinguished and a version that only includes invoiceable entries. Gross margin percentages are calculated as gross margins divided by the registered billing price.

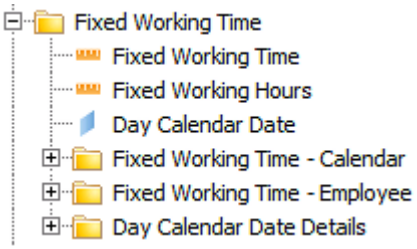
[GM] JOBENTRY.BILLINGPRICEREGBASE − JOBENTRY.COSTPRICEREG
[GM %] $\frac{\text{JOBENTRY.BILLINGPRICEREGBASE} - \text{JOBENTRY.COSTPRICEREG}}{\text{JOBENTRY.BILLINGPRICEREGBASE}}$ *if* JOBENTRY.BILLINGPRICEREGBASE ≠ 0 0 *otherwise*

[GM - Invoiceable]	JOBENTRY.BILLINGPRICEREGBASE − JOBENTRY.COSTPRICEREG *if* EXACTIVITY.TOBEINVOICEDPN = 0 *or* EXJOBHEADER.NOTINVOICEABLEPN = 1
[GM % - Invoiceable]	$\frac{\text{JOBENTRY.BILLINGPRICEREGBASE} - \text{JOBENTRY.COSTPRICEREG}}{\text{JOBENTRY.BILLINGPRICEREGBASE}}$ *if* JOBENTRY.BILLINGPRICEREGBASE ≠ 0 *and* (EXACTIVITY.TOBEINVOICEDPN = 0 *or* EXJOBHEADER.NOTINVOICEABLEPN = 1) 0 *otherwise*

8.3.5 Fixed Working Time

Objects for reporting on the fixed working time, are located in the [Fixed Working Time] folder.

Fixed working time can be calculated based on: (i) the **Day Calendar** assigned to employees and (ii) the revision of the individual employee. When calculating based on the **Day Calendar** the field DAYCALENDARLINE.FIXEDTIME is used. The field gives the fixed time for each date in the calendar. When calculating based on the employee revision, we need to know which day we need the fixed hours for as the table is denormalized. Thus, we get the fixed working hours from the fields FIXEDWORKTIMEMONDAY, FIXEDWORKTIMETUESDAY, ..., FIXEDWORKTIMESUNDAY of the table EMPLOYEEREVISION. These fields are associated the present date in the **Day Calendar** by means of the first date in the week and the number of the week day. Thereby, the fixed time from the employee revision and the fixed time from the **Day Calendar** are associated by date and we can thus compare them. In general, the fixed working time is just the largest value of the fixed time from the calendar and the employee revision. But there are individual objects for getting the fixed time as defined in the calendar and the employee revisions, respectively.



[Fixed Working Time]

$$max \begin{cases} \texttt{DAYCALENDARLINE.FIXEDWORKINGTIME} \\ FixedEmployeeTime \end{cases}$$

$$where\ FixedEmployeeTime =$$

$$\begin{aligned}
&\texttt{EMPLOYEEREVISION.FIXEDWORKINGTIMEMONDAY} \\
&\quad if\ \texttt{CALENDARDAY.DAYOFWEEK} = 1 \\
&\texttt{EMPLOYEEREVISION.FIXEDWORKINGTIMETUESDAY} \\
&\quad if\ \texttt{CALENDARDAY.DAYOFWEEK} = 2 \\
&\texttt{EMPLOYEEREVISION.FIXEDWORKINGTIMEWEDNESDAT} \\
&\quad if\ \texttt{CALENDARDAY.DAYOFWEEK} = 3 \\
&\texttt{EMPLOYEEREVISION.FIXEDWORKINGTIMETHURSDAY} \\
&\quad if\ \texttt{CALENDARDAY.DAYOFWEEK} = 4 \\
&\texttt{EMPLOYEEREVISION.FIXEDWORKINGTIMEFRIDAY} \\
&\quad if\ \texttt{CALENDARDAY.DAYOFWEEK} = 5 \\
&\texttt{EMPLOYEEREVISION.FIXEDWORKINGTIMESATURDAY} \\
&\quad if\ \texttt{CALENDARDAY.DAYOFWEEK} = 6 \\
&\texttt{EMPLOYEEREVISION.FIXEDWORKINGTIMESUNDAY} \\
&\quad if\ \texttt{CALENDARDAY.DAYOFWEEK} = 7
\end{aligned}$$

[Fixed Working Time]

$$max \begin{cases} \texttt{DAYCALENDARLINE.FIXEDWORKINGTIME} \\ FixedEmployeeTime \end{cases}$$
$$if\ \texttt{EXEMPLOYEEREVISION.TIMEREGISTRATIONUNITPN} = 0$$

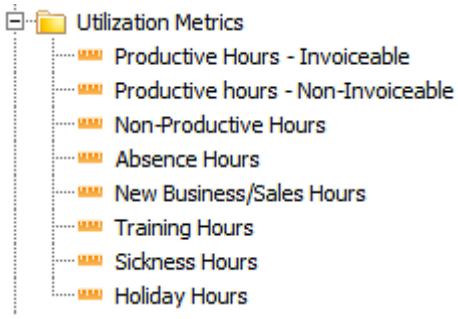
$$\texttt{EXEMPLOYEEREVISION.HOURSPERMANDAY} * max \begin{cases} \texttt{DAYCALENDARLINE.FIXEDWORKINGTIME} \\ FixedEmployeeTime \end{cases}$$
$$if\ \texttt{EXEMPLOYEEREVISION.TIMEREGISTRATIONUNITPN} = 1$$

$$where\ FixedEmployeeTime =$$
$$\texttt{EMPLOYEEREVISION.FIXEDWORKINGTIMEMONDAY}\ if\ \texttt{CALENDARDAY.DAYOFWEEK} = 1$$
$$\texttt{EMPLOYEEREVISION.FIXEDWORKINGTIMETUESDAY}\ if\ \texttt{CALENDARDAY.DAYOFWEEK} = 2$$
$$\texttt{EMPLOYEEREVISION.FIXEDWORKINGTIMEWEDNESDAT}\ if\ \texttt{CALENDARDAY.DAYOFWEEK} = 3$$
$$\texttt{EMPLOYEEREVISION.FIXEDWORKINGTIMETHURSDAY}\ if\ \texttt{CALENDARDAY.DAYOFWEEK} = 4$$
$$\texttt{EMPLOYEEREVISION.FIXEDWORKINGTIMEFRIDAY}\ if\ \texttt{CALENDARDAY.DAYOFWEEK} = 5$$
$$\texttt{EMPLOYEEREVISION.FIXEDWORKINGTIMESATURDAY}\ if\ \texttt{CALENDARDAY.DAYOFWEEK} = 6$$
$$\texttt{EMPLOYEEREVISION.FIXEDWORKINGTIMESUNDAY}\ if\ \texttt{CALENDARDAY.DAYOFWEEK} = 7$$

8.3.6 Utilization Metrics

On each activity, it is possible to state a **Utilization** popup literal. These popup literals are freely defined and named by users configuring the Maconomy system. It is done in the window **Popup Fields**. The Utilization universe assumes a specific set of such utilization popups; namely the ones defined by the **PSO** solution. The objects for reporting on registered time sliced by these assumed utilization popups, are located in the folder [Utilization Metrics].

For each utilization popup literal, there is an object that states the registered time on the activity having a specific utilization popup literal



[Productive Hours - Invoiceable]	JOBENTRY.NUMBERREGISTERED *if* EXACTIVITY.EMPLOYEEUTILIZATIONPN = 0
[Productive Hours - Non-Invoiceable]	JOBENTRY.NUMBERREGISTERED *if* EXACTIVITY.EMPLOYEEUTILIZATIONPN = 1
[Non-Productive Hours]	JOBENTRY.NUMBERREGISTERED *if* EXACTIVITY.EMPLOYEEUTILIZATIONPN = 2
[Absence Hours]	JOBENTRY.NUMBERREGISTERED *if* EXACTIVITY.EMPLOYEEUTILIZATIONPN = 3
[New Business/Sales Hours]	JOBENTRY.NUMBERREGISTERED *if* EXACTIVITY.EMPLOYEEUTILIZATIONPN = 4
[Training Hours]	JOBENTRY.NUMBERREGISTERED *if* EXACTIVITY.EMPLOYEEUTILIZATIONPN = 5
[Sickness Hours]	JOBENTRY.NUMBERREGISTERED *if* EXACTIVITY.EMPLOYEEUTILIZATIONPN = 6
[Holiday Hours]	JOBENTRY.NUMBERREGISTERED *if* EXACTIVITY.EMPLOYEEUTILIZATIONPN = 7

R The objects in the [Utilization Metrics] folder all assume a specific setup of utilization popups. If other popups are needed, the objects may have another meaning as the objects use the internal enumeration value to distinguish the popups. Thus the first utilization popup could have been set up to mean non-invoiceable hours and the activity could be set to non-invoiceable. But the first object in the folder is named to indicate that it is invoiceable hours. Therefore, it is important to customize the objects in this folder so that they match the setup of the utilization popups.

8.4 Reporting Examples

In principle, most utilization reports are alike as they display measures related to some performance of employees and associate to employee dimensions like location or entity. However, what performance means is where the utilization reports really differ.

Some utilization reports are centered on categorizing registration after the employee utilization associated the activity of the registrations. Other utilization reports compare figures against fixed hours to highlight over-performance.

In general, we can say that utilization has to do with employee performance related to individual goals set up for the employee. However, choosing the right sorts of goals highly influences how useful and how fair the report's conclusion is. Measuring invoiceable hours is e.g. unfair in organizations consisting of both internal development work and work related directly to customers. In the first category, many hours may not even be invoiceable.

In the following, we shall just give a few examples of some simple utilization reports. Whether they are relevant for a given organization is a totally other matter. Utilization reports really rely on having the right employee utilization setup and measures for the reports. The measures in the Utilization universe are general enough but only a fixed set of dedicated employee utilization measures (from the PSO setup) are included. However, many other measures can be defined from the general measures by utilizing the available dimensions in the universe. It should thus be possible to define all the necessary sorts of measures as variables in utilization reports.

Example 8.1 (PSO employee utilization report) *In this example, we shall create a simple utilization report that utilizes the PSO specific utilization measures. These measures are assuming the PSO setup of* ***Employee Utilization*** *for activities.*

1. *Create a new WebIntelligence report and choose the universe* Utilization.
2. *Add the following objects to the query:*

   [Company] *(folder [Report])*

   [Location] *(folder [Report])*

   [Employee No.] *(folder [Employee])*

   [Employee Name] *(folder [Employee])*

   [Productive Hours - Invoiceable] *(folder [Employee Utilization])*

   [Productive Hours - Non-Invoiceable] *(folder [Employee Utilization])*

   [Non-Productive Hours] *(folder [Employee Utilization])*

   [Absence Hours] *(folder [Employee Utilization])*

[Holiday Hours] *(folder [Employee Utilization])*

3. *Drag the objects [Company] and [Location] up above the table to create sections.*

■

Example 8.2 (Comparing registered to fixed hours) *In this example, we shall create a simple utilization report that outlines the registered against fixed hours to illustrate the utilization of employee's available time.*

1. *Create a new WebIntelligence report and choose the universe* Utilization.
2. *Add the following objects to the query:*

   Company *(folder [Report])*

   Location *(folder [Report])*

   Employee No. *(folder [Employee])*

   Employee Name *(folder [Employee])*

   Fixed Working Hours *(folder [Fixed Working Time])*

   Productive Hours *(folder [Registered])*

   Productive Invoiceable Hours *(folder [Invoiceable])*

3. *Drag the objects [Company] and [Location] up above the table to create sections.*

■

Chapter 9

Finance Universe

The Finance universe contains objects for reporting on financial figures in Maconomy. It is possible to report on both **Actuals** and **Budget** figures as well as **Opening balances** and **Closing Balances**. It is possible to distinguish between **Profit and Loss Accounts** and **Balance Accounts**, as well as designating figures on the **Year-end-Result account**. Thereby, it is possible to provide the foundation for the different classical reports like **Balance Sheet**, **Profit and Loss**, and **Trial Balance**. It is also possible to report on budgets and compare these to the actuals on the same dimensions (e.g. account and fiscal period). When reporting on budgets, budget models are used for designating budgets of individual fiscal years.

It is possible to associate figures to levels in **Reporting Structures** in order to organize accounts in a hierarchy or filter accounts for the specific reporting purposes. E.g., it is possible to define cashflow reports by limiting the set of accounts to exactly those that the company needs for that kind of analysis.

The measure objects can be grouped and restricted by a variety of dimensions including company, account, customer, vendor, job, employee, fiscal periods, dates and the ten standard dimensions.

The universe also supports reporting on **Local Charts of Accounts**. When including account or reporting structure objects in reports, the user will be prompted to select whether to view data structured after the global chart of accounts or local charts of accounts.

9.1 Prerequisites

In order to fully benefit from the Finance universe, Maconomy should be set up such that:

- The system parameter **Use Calendar Day** is enabled and the calendar spans all

necessary dates.

- The necessary reporting structures are defined.

9.2 Maconomy Workflows

In the following sub-sections we shall look at the common Maconomy workflows that are concerned with the data for which the Finance universe is made for reporting on. First we shall look at a typical journal posting workflow and see which records that are produced in that process. Then we shall consider financial budgeting and then how to hierarchically structure accounts using **Reporting Structures**.

9.2.1 Finance Postings

In Maconomy, financial postings are done in all workflows that concern business data that somehow end up on the general ledger. In some cases, this happens as a side effect that is not presented to the user. An example is that financial postings are done when printing an invoice or registering a customer payment. In other cases, the financial posting is done directly in Maconomy windows like **General Journal**. In both kinds of scenarios, the result is the creation of records in the `FINANCEENTRY` table.

In all financial postings, changes are also happening in two additional tables: `DIMENSIONPERIOD` and `FINANCEPERIOD`. These tables are aggregated tables of the `FINANCEENTRY` table. The records in the `DIMENSIONPERIOD` table group finance entries by account, company, fiscal period and the ten standard dimensions. When a financial posting is done to an account (also producing records in the `FINANCEENTRY` table), a record in the `DIMENSIONPERIOD` table is either created or updated. A record is created if no posting has been done so far to the same combination of account, company, fiscal period and standard dimensions; otherwise, a record is updated. In that case the fiscal amounts (e.g. the debit and credit figures) are added to the figures already registered on the `DIMENSIONPERIOD` record.

The principle is the same for the `FINANCEPERIOD` table, except that it does not group on standard dimensions. So all finance figures posted to the same combination of account, company, and fiscal period, update the same record in the `FINANCEPERIOD` table.

Assume the existence of a job and registrations on that job:

1. In the window **Invoice Selection** approve the invoice selection.
2. Print the invoice.

   This provides—depending on setup and the kind of registration and invoice issued—typically four finance entries: One on the customer control account, one on open cost, one on open billing, and one on the WIP account. Correspondingly records in `DIMENSIONPERIOD` and `FINANCEPERIOD` are created or updated.

Now consider the following other scenario:

1. In the window **General Journal** create a new journal.
2. In the lower panel, create a row with **GRP**=`General`.
3. Select an account and enter a debit amount.
4. As offset choose "Cash".

   This results in the creation of a finance entry on the selected account and a corresponding finance entry on the offset account (here, one of the cash accounts). Correspondingly, it results in records or updates of records in the `DIMENSIONPERIOD` *and* `FINANCEPERIOD` *tables.*

R That `DIMENSIONPERIOD` and `FINANCEPERIOD` are summations of `FINANCEENTRY` always, is an incomplete truth. The figures calculated on the fields of the `DIMENSIONPERIOD` and `FINANCEPERIOD` tables, assume the opening balance stored on the records of these tables. If **Year-End-Closing** has not been performed in Maconomy, the opening balances of the new year are not calculated in Maconomy and thus the summation tables are not updated. However, this is well-known by all customers familiar with Maconomy and the opening balances are almost always updated in practice when using Maconomy the right way. The condition is the exact same for the old finance reports in Maconomy (i.e. non-BPM).

9.2.2 Finance Budgeting

Financial budgets are typically done by creating budget journals. Each **Budget Journal** associates one or more entries with budget figures for each of the 12 possible fiscal periods (figures may be zero). When a **Budget Journal** has been posted, it results in updates of records in the table `BUDGETSUM`. This table has similarities with the `DIMENSIONPERIOD` table in that it groups budget figures by account, company, fiscal period, and standard dimensions. Typically, a more fine–grained level is not needed and typically the amount of budget entries is far smaller than the amount of finance entries. Hence, the only table to be concerned with in BPM Reporting for financial budgets, is the `BUDGETSUM` table.

In BPM, we usually identify financial budgets by **Budget Models**. This is the approach for all standard finance reports that display budget figures. Alternatively, a budget type and revision can be used. The Finance universe supports both approaches but standard finance reports only supports budget models.

Consider the following:

1. In the window **Budget Models** create a new budget model for the fiscal year.
2. In the window **Budget Journal** create a new journal on the new budget model.

3. In the lower panel, create 2 rows with a figure for each of the 12 fiscal months. The accounts for the two lines should not be the same but the 2 amounts for each fiscal period must summarize to zero.

4. Post the budget journal.

   This results in creation or update of two records in the `BUDGETSUM` *table; one for each account.*

9.2.3 Reporting Structures

When reporting on financial figures, we often want to group them on headers in the account structure. E.g., we might want to group profit and loss, assets, etc. We may also want to group on sub-groups; thereby dividing e.g. profit and loss into sub-categories of accounts for profit and loss. To support this when reporting, **Reporting Structures** are used. Each reporting structure consists of a collection of lines. Each line states an account number and has 12 fields for stating the names of the groups, sub-groups, sub-sub-groups, etc. that are associated the account. Each reporting structure is represented by a record in the table `REPORTINGSTRUCTUREHEADER` and each of the lines mentioned is a record in the table `REPORTINGSTRUCTURELINE`. Reporting structures can be identified uniquely by their name. However, they can also be identified uniquely by the following properties[1]:

- The type of the reporting structure (e.g. `Account`)
- Option List 1
- Selected Option 1
- From/To Date interval

The above means that there can only be one reporting structure on the same combination of type, option list, selected option and from/to date interval.

In the Finance universe, only reporting structures of the type `Account`, are used. The option list and selected value are used for distinguishing the different reporting structures that may be setup for financial reporting. One reporting structure (identified by one selected option) may be a standard one for most Balance Sheets and Profit & Loss reports. Another reporting structure may be used for cashflow statements, and so on.

For each reporting structure on a specific type and with a specific option list and selected option, it is furthermore possible to have multiple structures depending on date intervals. Thus, in one period, one reporting structure may apply and in the next, another one applies. This, of course, has to be handled in the filters of the report utilizing it. Thus, a finance report may prompt the user for a fiscal period and the corresponding start date of that period can then be used in a restriction on the From/To Date interval of the reporting structure.

[1] We shall later see that these—as fields—are more convenient to use for selecting a specific reporting structure.

Reporting structures are created as follows:

1. In the window **Reporting Structures**, create a new reporting structure of type `Account`.
2. In the lower panel, create the desired headers for the structure to be defined. Notice that headers can have sub-headers (i.e. sub-groups), etc.
3. Apply the action `Add Missing` to fill in accounts. In the wizard appearing, select an interval of accounts and state the group in which they should be inserted.
4. Continue until all desired accounts are added. Possibly, move accounts around, add new groups or sub-groups, etc. until the desired structure is achieved.

R Standard BPM finance documents create sub-total texts from the reporting structure headers by appending the text `Total` to them. Therefore, it is not recommended to include the term `Total` in the reporting structure headers.

The choice of header titles is in principle free but should support how the reporting structure is to be used. If it is used for grouping accounts, the grouping levels should be reasonable names for these account groups. E.g. the label *Revenue* could be used for grouping all accounts to which revenue is posted. If it is used for filtering and identifying types of accounts, the labels should likewise have reasonable names but in this case the exact text strings are required to match assumed labels. E.g. we could assign the label *CASHFLOW* to the first grouping level for accounts that we wish to include in cashflow reports. Notice the uppercase in order to avoid confusion of which characters should be upper or lower case. Pure lowercase could also be used.

9.2.4 Local Charts of Accounts

The universe also supports the ability to report on local charts of accounts, as an alternative to reporting on normal (global) financial accounts. Local charts of accounts are set up in the windows **Local Account Information Card** and **Local Charts of Accounts**. In the former, the individual local accounts are defined. In the latter, they are organize in local charts of account lists. Reporting structures should be set up for local charts of accounts, just as done for global accounts.

When reporting on accounts and reporting structures, the universe prompts the user for selecting whether to use global or local charts of accounts. Choosing global charts of accounts means that accounts from the table `ACCOUNT` are used. If choosing local charts of accounts, it means that accounts from the table `LOCALACCOUNT` are used. Local accounts are local to the individual company.

9.3 Business Layer

The universe provides measures for reporting on all sorts of finance actuals and budgeting figures posted in Maconomy. It is possible both to report on the individual debit and credit figures, balances, and tax amounts. It is also possible to report on opening and closing balances according to fiscal periods. The business layer contains a main folder for each of these three sorts of measures:

Actuals containing objects displaying financial postings as stored on *finance entries*, *dimension periods*, and *finance periods*

Budget containing objects displaying finance budget figures as stored on *budget sum* records.

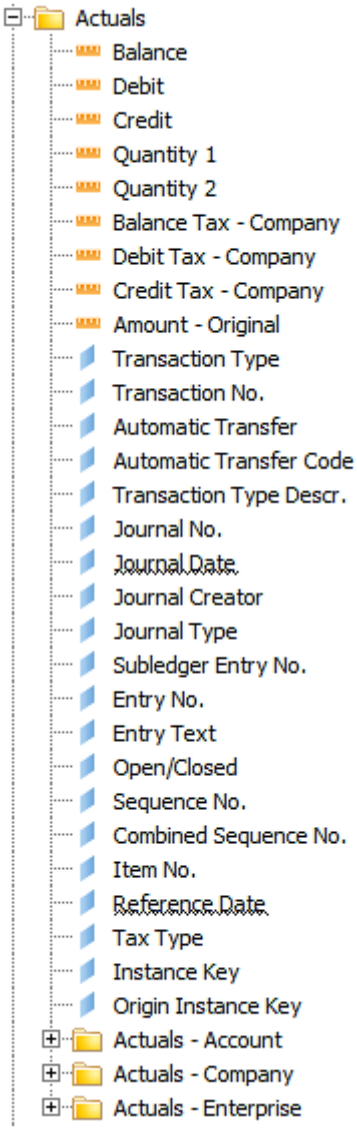
Opening/Closing Balances containing objects displaying opening and closing balances of fiscal periods.

In the following sub-sections, we shall go through each of these folders and explain the measure objects contained.

9.3.1 Actuals

Objects for reporting on financial actuals are located in the [`Actuals`] folder. The folder contains objects for reporting on the debit, credit, balance, quantities, and tax amounts of financial postings. For some of the amount objects, the user is prompted for selecting the currency type in which the figures are displayed. For other amount objects the figures will be displayed in a specific currency type. These objects are stored in sub-folders of the [`Actuals`] folder.

The measures are rooted in the fields of the `FINANCEENTRY` table. We utilize *Aggregate Awareness* to the tables `DIMENSIONPERIOD` and `FINANCEPERIOD`. The former measure objects are used if not involving other dimensions than company, account, standard dimensions, and finance periods. The latter measure objects are used if not involving other dimensions than company, account, and fiscal period.



For simplicity, we shall specify the objects assuming that the currency type is **Company Currency**. We state the definitions of the objects assuming the `FINANCEENTRY` level. Definitions are similar when on the levels of `DIMENSIONPERIOD` and `FINANCEPERIOD`.

[Balance - Company] FINANCEENTRY.DEBIT − FINANCEENTRY.CREDIT
[Debit - Company] FINANCEENTRY.DEBIT
[Debit - Company] FINANCEENTRY.CREDIT

[Quantity 1]	FINANCEENTRY.QUANTITYA
[Quantity 2]	FINANCEENTRY.QUANTITYB
[Balance Tax - Company]	FINANCEENTRY.DEBITVAT − FINANCEENTRY.CREDITVAT
[Debit Tax - Company]	FINANCEENTRY.DEBITVAT
[Credit Tax - Company]	FINANCEENTRY.CREDITVAT

When the tables DIMENSIONPERIOD and FINANCEPERIOD are used, we have to "lookup" the field to use by means of the present finance period. The reason is that—contrary to the FINANCEENTRY table— the two tables have fields corresponding to the 12 fiscal periods. Thus, if the fiscal period is the *first*, we need to fetch e.g. debit and credit figures from the fields DIMENSIONPERIOD.DEBIT1 and DIMENSIONPERIOD.CREDIT1, and FINANCEPERIOD.DEBIT1 and fINANCEPERIOD.CREDIT1, respectively. This means that the tables DIMENSIONPERIOD and FINANCEPERIOD are not *denormalized*.

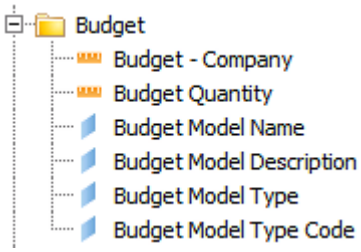
This also influences the calculation of Tax when using the DIMENSIONPERIOD or FINANCEPERIOD tables. The Tax is not given by **Debit Tax** minus **Credit Tax**, but by a single field DIMENSIONPERIOD.VAT$_i$ and FINANCEPERIOD.VAT$_i$; where i is a fiscal period.

9.3.2 Budget

Objects for reporting on financial budget figures, are located in the [Budget] folder. The folder contains objects for reporting on the budgeted amount and quantity for fiscal periods. Finance budgets are only available in the currency of the company.

R Notice that because budget figures are only available in the currency of the company, it is important to not compare with actuals in other currency types.

Objects for reporting on budget figures, are taken from the normalized table BUDGETSUM which corresponds—level-wise—to DIMENSIONPERIOD. Budget figures are only available in Maconomy in the currency of the company. Thus, there is just one measure object and one object for budgeted quantity. Both are given by the fiscal period as it is the case for actuals.



In the following table, i denotes the fiscal period.

[Budget - Company]	BUDGETSUM.AMOUNTBASE
[Budget Quantity]	BUDGETSUM.QUANTITYA

R Maconomy does not provide a QUANTITYB field for budgets as it is the case for actuals.

9.3.3 Opening and Closing Balances

Objects for reporting on opening and closing balances are located in the [Opening/Closing Balances] folder. Opening and closing balances are always associated fiscal periods and thus should be restricted by such periods.

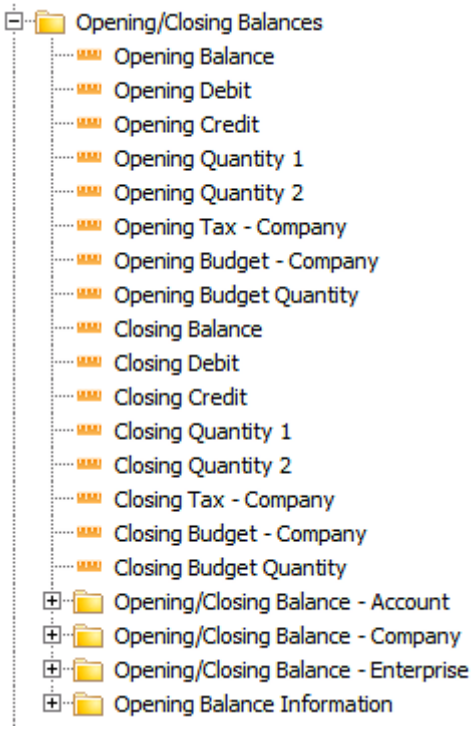
This is also the reason why opening and closing balances are only defined on the level of DIMENSIONPERIOD and FINANCEPERIOD; not on the level of FINANCEENTRY.

The opening balance is the opening balance of a fiscal period. If we thus restrict to a certain fiscal period, it will be the opening balance of that period. If we combine such a measure object with one of the fiscal period objects, we get the opening balance for each of the available fiscal periods. Similar goes for closing balances.

R Opening and closing balances state the balance as of a given fiscal period. It is important to restrict to a specific fiscal period when using these objects. Hence, it does not make sense to summarize opening and closing balances across fiscal periods; nor for an interval of periods.

Due to the *denormalized* nature of the tables DIMENSIONPERIOD and FINANCEPERIOD (see Section 9.3.1, we cannot fetch the opening or closing balance by summarizing over a single database field. We need to lookup the opening balance of the fiscal year and then add the debit and credit adjustments from the individual fiscal periods up to the period of interest. Thus, if we wish the opening balance of the third fiscal period, we must take the opening balance of the first fiscal period and add the balances (i.e. movements) of the first and second fiscal period. This logic is handled by the measure objects.

There are opening and closing balance objects for the same sorts of figures as we have other finance measure objects. That is, the three currency types *Account Currency*, *Company Currency*, and *Enterprise Currency*. The folder [`Opening Balance Information`] contains special objects for reporting on opening balances in a way optimised for performance.



For simplicity, we shall in the following assume the objects defined on the fields of `FINANCEPERIOD` and in company currency. In the following table n denotes the number of the fiscal period for which we define opening and closing balances.

[`Opening Balance - Company`] $\texttt{FINANCEPERIOD.DEBITOPENING} - \texttt{FINANCEPERIOD.CREDITOPENING} + \sum_{i=1}^{n-1} \texttt{FINANCEPERIOD.DEBIT}_i - \sum_{i=1}^{n-1} \texttt{FINANCEPERIOD.CREDIT}_i$ *where* n *is the fiscal period of interest.*
[`Opening Debit - Company`] $\texttt{FINANCEPERIOD.DEBITOPENING} + \sum_{i=1}^{n-1} \texttt{FINANCEPERIOD.DEBIT}_i$ *where* n *is the fiscal period of interest.*
[`Opening Credit - Company`] $\texttt{FINANCEPERIOD.CREDITOPENING} + \sum_{i=1}^{n-1} \texttt{FINANCEPERIOD.CREDIT}_i$

where n *is the fiscal period of interest.*

[Opening Quantity 1]

$$\texttt{FINANCEPERIOD.QUANTITYAOPENING} + \sum_{i=1}^{n-1} \texttt{FINANCEPERIOD.QUANTITYA}_i$$

where n *is the fiscal period of interest.*

[Opening Quantity 2]

$$\texttt{FINANCEPERIOD.QUANTITYBOPENING} + \sum_{i=1}^{n-1} \texttt{FINANCEPERIOD.QUANTITYA}_i$$

where n *is the fiscal period of interest.*

[Opening Tax - Company]

$$\texttt{FINANCEPERIOD.OPENINGVAT} + \sum_{i=1}^{n-1} \texttt{FINANCEPERIOD.VAT}_i$$

where n *is the fiscal period of interest.*

[Opening Budget - Company]

$$\texttt{BUDGETSUM.AMOUNTBASEOPENING} + \sum_{i=1}^{n-1} \texttt{BUDGETSUM.AMOUNTBASE}_i$$

where n *is the fiscal period of interest.*

[Opening Budget Quantity]

$$\texttt{BUDGETSUM.QUANTITYOPENING} + \sum_{i=1}^{n-1} \texttt{BUDGETSUM.AMOUNTBASE}_i$$

where n *is the fiscal period of interest.*

[Closing Balance - Company]

$$\frac{\texttt{FINANCEPERIOD.DEBITOPENING} - \texttt{FINANCEPERIOD.CREDITOPENING} + \sum_{i=1}^{n} \texttt{FINANCEPERIOD.DEBIT}_i}{- \sum_{i=1}^{n} \texttt{FINANCEPERIOD.CREDIT}_i}$$

where n *is the fiscal period of interest.*

[Closing Debit - Company]

$$\texttt{FINANCEPERIOD.DEBITOPENING} + \sum_{i=1}^{n} \texttt{FINANCEPERIOD.DEBIT}_i$$

where n *is the fiscal period of interest.*

[Closing Credit - Company]

$$\texttt{FINANCEPERIOD.CREDITOPENING} + \sum_{i=1}^{n-1} \texttt{FINANCEPERIOD.CREDIT}_i$$

where n *is the fiscal period of interest.*

[Closing Quantity 1] $\texttt{FINANCEPERIOD.QUANTITYAOPENING} + \sum_{i=1}^{n} \texttt{FINANCEPERIOD.QUANTITYA}_i$ *where* n *is the fiscal period of interest.*
[Closing Quantity 2] $\texttt{FINANCEPERIOD.QUANTITYBOPENING} + \sum_{i=1}^{n} \texttt{FINANCEPERIOD.QUANTITYA}_i$ *where* n *is the fiscal period of interest.*
[Closing Tax - Company] $\texttt{FINANCEPERIOD.OPENINGVAT} + \sum_{i=1}^{n} \texttt{FINANCEPERIOD.VAT}_i$ *where* n *is the fiscal period of interest.*
[Closing Budget - Company] $\texttt{BUDGETSUM.AMOUNTBASEOPENING} + \sum_{i=1}^{n} \texttt{BUDGETSUM.AMOUNTBASE}_i$ *where* n *is the fiscal period of interest.*
[Closing Budget Quantity] $\texttt{BUDGETSUM.QUANTITYOPENING} + \sum_{i=1}^{n} \texttt{BUDGETSUM.AMOUNTBASE}_i$

R Notice, that the only real difference between calculating an *opening* and a *closing* balance is whether we include the figure of the present fiscal period.

R Also notice how calculations of opening and closing figures are relying on the opening balances of a fiscal year. These balances are fields on the denormalized tables `FINANCEPERIOD` and `DIMENSIONPERIOD`, and are maintained by Maconomy when closing a fiscal year.

9.4 Reporting Examples

9.4.1 Working with debit, credit and balance objects

The debit, credit and balance objects are the most fundamental measure objects of the Finance universe. Typically, we want to combine them with dimensions like account, company, currency, etc., and then filter on fiscal period; either a specific year and period, or some sort of range.

In the following, we shall consider some simple G/L reports that list financial data of Maconomy. We shall build up from the very simple to more complex reports.

Example 9.1 (A simple account enquiry) *In this example, we shall create a simple account enquiry report which is to form the basis for some of the succeeding finance reports.*

1. *Create a new WebIntelligence document and select the universe* Finance *as data provider.*
2. *Select the following objects for the query:*

   [Account No.] *(folder [Account])*

   [Account Name] *(folder [Account])*

   [Debit] *(folder [Actuals])*

   [Credit] *(folder [Actuals])*
3. *for the report filter, drag in the object [Fiscal Year-Period] (folder [Fiscal Period]. Define it as a prompt.*
4. *Run the report.*

In addition to the prompt for selecting a fiscal period, the report will prompt for choosing whether to use **Global** *or* **Local** *account structures, and in which currency type (company currency, account currency or enterprise currency) the figures should be displayed. The figures displayed will be those posted within the select fiscal year and period. E.g., if selecting 2016-01, we see the figures posted in the first financial period of year 2016. If several postings have been done to the same account (and that is typically the case), debit amounts are summarized by account, and likewise for credit amounts.* ■

If we inspect the generated SQL of the report from the example above, we see that data are fetched from FINANCEPERIOD_D which is defined on the database table FINANCEPERIOD. This is the most aggregated finance table which is suitable for performance.

Example 9.2 (Account enquiry by standard dimension) *We shall enhance the report document developed in the previous example. Add the object [Location Name] (folder [Dimensions]/[Location]). Then run the report again and investigate the generated SQL.* ■

The difference compared to the previous example, is that now data are fetched from DIMENSIONPERIOD_D which is defined on DIMENSIONPERIOD. The aggregate awareness ensures that this table is used because we added the location dimension. This makes it impossible to use the FINANCEPERIOD table so DIMENSIONPERIOD is the second best we can do.

The two previous examples showed how the inclusion of dimensions changes the level of aggregation from which we fetch finance data. In the following example, we shall examine

the third and lowest level. This level is the level of finance entries which is not aggregated. It is also the level where performance may be challenged because `FINANCEENTRY`—by far—is the largest table in Maconomy.

Example 9.3 (Date–filtered account enquiry) *In this example we shall create a report similar to the one in the previous example. However, instead of filtering by fiscal period, we shall limit by entry date.*

1. *Create a new WebIntelligence document*
2. *Add the following objects:*

   [Account No.] *(folder [Account])*

   [Account Name] *(folder [Account])*

   [Debit] *(folder [Actuals])*

   [Credit] *(folder [Actuals])*
3. *for the report filter, drag in the object [Entry Date] (folder [Entry Date]. Define it as using two prompts: a* from *and a* to *date prompt using the* **between** *operator.*
4. *Run the report.*

Again, investigate the generated SQL. We see that data are now fetched from the `FINANCEENTRY` table.

Example 9.4 (Simple Profit and Loss report) *In this example, we shall create a simple profit and loss report. For this report we shall be using the opening and closing measure objects which show the opening balance and the closing balance of a selected fiscal period.*

1. *Create a new WebIntelligence document*
2. *Add the following objects:*

   [Currency] *(folder [Currency])*

   [Company No.] *(folder [Company Information])*

   [Account No.] *(folder [Account])*

   [Account Name] *(folder [Account])*

   [Opening Balance] *(folder [Opening / Closing Balances])*

   [Closing Balance] *(folder [Opening / Closing Balances])*
3. *for the report filter, drag in the object [Fiscal Year-Period] as a prompt and*

the filter [P&L Accounts]

4. *Run the report*

This gives us a simple profit and loss report. Drag the [Currency] and [Company No.] objects out as sections. This way we ensure that figures from different companies (which could use different fiscal year templates and company currencies) and different currencies are not mixed. The report only includes profit and loss accounts and will show the opening and closing balance of the selected fiscal year and period. ■

9.4.2 Accounts and reporting structures

In the above, we have not been concerned with ordering the accounts properly or grouping them. This is best done by the user of a reporting structure set up in Maconomy. In the following, we shall add this kind of structure to the reports we create.

Example 9.5 *In this example, we shall elaborate on the report document we created in Example 9.4.*

1. *Add the following objects to the query:*

   [Grouping Level 1] *(folder [Account / Account Grouping])*

   [Grouping Level 2] *(folder [Account / Account Grouping])*

   [Global Sorting] *(folder [Account / Account Grouping])*

2. *Add the following objects to the filter:*

   [Option List Number 1] *(folder [Account / Account Grouping])*

   [Selected Option 1] *(folder [Account / Account Grouping])*

3. *Assign the restriction: [Option List Number 1] equals Finance*

4. *Assign the restriction: [Selected Option 1] equals Standard*[a]

5. *In the report, add the new objects as columns to the report table.*

6. *Drag the two grouping level objects up above the table to create sections.*

7. *Sort the rows in the table by the [Global Sorting] object.*

8. *Right-click the [Global Sorting] column and select* Hide *and then* Hide Dimension.

9. *Refresh the report*

■

[a] If another reporting structure is to be used, the option list and selected option restrictions should reflect that. Also, if date dependent reporting structures are defined, an additional filter on the date range of the reporting structure, must be applied. This is outside the scope of this example.

9.4.3 Utilizing the zero line context

In the previous examples, the reports only listed accounts for which postings have been done. However, often we wish to list all accounts regardlessly of any postings. This means that we wish to include **Zero-Lines**. This is done by splitting into a separate account query including the object [Show Zero Lines], and a query selecting the amounts. Both queries should share dimensions that can be merged.

Example 9.6 *In this example, we shall create a simple* ***Balance Sheet*** *report which utilizes the zero-line principle.*

1. *Select the following objects for a query* Accounts*:*

   [Currency] *(folder [Currency])*

   [Company No.] *(folder [Company Information])*

   [Account No.] *(folder [Account])*

   [Account Name] *(folder [Account])*

   [Grouping Level 1] *(folder [Account / Account Grouping])*

   [Grouping Level 2] *(folder [Account / Account Grouping])*

   [Global Sorting] *(folder [Account / Account Grouping])*

   [Zero Lines Context] *(folder [Report])*

2. *for the report filter, select:*

   [Balance Accounts] *(folder [Account]*

   [Fiscal Year-Period] *(folder [Fiscal Period]*

   [Option List Number 1] *(folder [Account / Account Grouping])*

   [Selected Option 1] *(folder [Account / Account Grouping])*

3. *Assign the restriction: [Option List Number 1] equals Finance*

4. *Assign the restriction: [Selected Option 1] equals Standard*

5. *Select the following objects for a query* Balances*:*

   [Currency] *(folder [Currency])*

   [Company No.] *(folder [Company Information])*

   [Account No.] *(folder [Account])*

   [Account Name] *(folder [Account])*

   [Grouping Level 1] *(folder [Account / Account Grouping])*

[Grouping Level 2] *(folder [Account / Account Grouping])*

[Global Sorting] *(folder [Account / Account Grouping])*

[Opening Balance] *(folder [Opening/Closing Balances])*

[Closing Balance] *(folder [Opening/Closing Balances])*

6. *for the report filter, select:*

   [Balance Accounts] *(folder [Account]*

   [Fiscal Year-Period] *(folder [Fiscal Period]*

   [Option List Number 1] *(folder [Account / Account Grouping])*

   [Selected Option 1] *(folder [Account / Account Grouping])*

7. *Assign the restriction: [Option List Number 1] equals Finance*
8. *Assign the restriction: [Selected Option 1] equals Standard*
9. *Merge the following dimension objects:*

   [Currency]

   [Company No.]

   [Account No.]

   [Account Name]

   [Grouping Level 1]

   [Grouping Level 2]

   [Global Sorting]

10. *Drag in the merged dimensions and the measure objects into the report in a table.*
11. *Drag the [Grouping Level 1] and [Grouping Level 2] to form sections.*
12. *Sort by the [Global Sorting] object and then hide that dimension.*
13. *Refresh the report.*

The report will—in addition to the reports we have seen so far—also prompt for whether to show zero lines. If answering No*, only accounts for which there are postings are shown. If answering* Yes*, all accounts are shown.*

However, the present report is not a fully functional balance sheet report. The reason is that a balance sheet report should include the so-called ***Retained Earnings****. We shall see how this is handled in the following section.* ■

9.4.4 Year-end Result Account

The balance sheet we created in the previous section lacks *retained earnings*. On the **Year-End Result Account**—as setup in Maconomy—we need to summarize last years result as well as current P&L balance. We do so by correcting the report to also query the Year-End Result Account.

Example 9.7 *We shall elaborate on the balance sheet report we created in Example 9.6. In this example, we shall complete the report by adding* ***Retained Earnings****.*

1. *Turn the query* `Balances` *into a combined query.*
2. *For this query select the objects:*

   `[Currency]` *(folder [*`Currency`*])*

   `[Company No.]` *(folder [*`Company Information`*])*

   `[Year End Result Account No.]` *(folder [*`Account / Year End Result Account`*])*

   `[Year End Result Account Name]` *(folder [*`Account / Year End Result Account`*])*

   `[Grouping Level 1]` *(folder [*`Account / Account Grouping`*])*

   `[Grouping Level 2]` *(folder [*`Account / Account Grouping`*])*

   `[Global Sorting]` *(folder [*`Account / Account Grouping`*])*

   `[Opening Balance]` *(folder [*`Opening/Closing Balances`*])*

   `[Closing Balance]` *(folder [*`Opening/Closing Balances`*])*
3. *Add the same filter objects and restrictions as in the existing* `Balances` *query; however, replace the filter [*`Balance Accounts`*] with [*`P&L Accounts`*].*
4. *Refresh the report.*

We see that the year-end result account is now included and thereby we have a complete balance sheet report. ■

9.5 Data Foundation

9.5.1 Fact Tables

The universe is centered on two sorts of facts:

Finance actuals which concern posted amounts to the general ledger. Amounts are captured on three different levels:

`FINANCEENTRY` which is the lowest level.

`DIMENSIONPERIOD` which aggregates finance entries by company, account, fiscal period and standard dimensions.

`FINANCEPERIOD` which aggregates dimension periods by company, account and fiscal period.

Budgeted amounts which concern budget figures posted in Maconomy's finance module.

In addition, a special fact `ZEROLINE_D` is present. This table is not a fact in the sense of providing foundation for measure objects. Rather it serves as binding between certain dimensions so that it is possible to report on accounts without requiring that postings have been done to the accounts.

In the following sections we shall explain these table further. In addition, we shall explain additional special features about the data foundation.

FINANCEPERIOD_D

This fact table is based on `FINANCEPERIOD`. It is the most aggregated table and is utilized only combining measures with company, account, fiscal period and currency dimensions.

R It is strongly recommended that finance reports are written so that they utilize this most aggregated level. This can be achieved by using restriction on fiscal periods rather than dates or date intervals, and avoiding restricting on dimensions like customer, vendor, standard dimensions and transactional details like transaction type. Filters can easily be applied on such dimensions though. If making these filters with optional prompts, they only break the aggregate awareness with users do apply restrictions.

The fact table is joined to the following 1. level dimension tables:

`CURRENTDATE_D` which provides the ability of combining with the current date. However, this does not mean that data can be sliced down on that level of detail. But it makes is possible to utilize current date for various purposes in reports.

`CURRENCY_D` which provides the ability of associating the currencies of the account, company and enterprise with the data from the fact.

`ACCOUNT_D` which provides the ability of associating accounts and account groups (**Reporting Structures**) to the data of the fact.

`COMPANYINFORMATION_D` which provides the ability of associating company information to the data of the fact.

`EXFISCALYEARPERIOD` which provides the ability of associating fiscal period information to data of the fact. This table is further joined to a snow-flake dimension `CURRENT-`

FISCALYEARPERIOD_D for associating the current fiscal period to the data of the fact.

DIMENSIONPERIOD_D

This fact table is based on DIMENSIONPERIOD which is the second most aggregated table. It should be used if it is not possible to use the FINANCEENTRY_D table.

The fact table is joined to the same tables as FINANCEENTRY_D is. In addition, it joins to all 10 standard dimensions.

FINANCEENTRY_D

This fact table is based on the FINANCEENTRY table and is the lowest level. It is not aggregated. The table is joined to all 1. level dimensions. These include all the tables mentioned for FINANCEPERIOD_D, standard dimensions, and in addition:

PAYMENTCUSTOMER which is an alias of COMPANYCUSTOMER_D and provides the ability of reporting on payment customer data related to the fact.

COMPANYCUSTOMER_D which provides the ability of reporting on responsible customer data related to the fact.

COMPANYVENDOR_D which provides the ability of reporting on vendor data related to the fact.

JOBHEADER_D which provides the ability of reporting on job data related to the fact.

BOACTIVITY which provides the ability of reporting on activity data related to the fact.

TASKLISTLINE_D which provides the ability of reporting on task data related to the fact.

BOEMPLOYEE which provides the ability of reporting on employee data related to the fact.

ENTRYDATE which provides the ability of reporting on entry date date related to the fact.

POSTINGDATE which provides the ability of reporting on posting date date related to the fact.

ASSET_D which provides the ability of reporting on asset information related to the fact.

BUDGETSUM_D

This fact table is based on the table BUDGETSUM which aggregates finance budget figures in Maconomy. The aggregate level is the same as that of DIMENSIONPERIOD. That is, it aggregates on companies, accounts, fiscal periods, and currencies.

R Note, that it is not possible to break down finance budget data on e.g. dimensions like customer and vendor. That level of detail does not exist in the table BUDGETSUM of Maconomy.

ZEROLINE_D

Common to all universes of BPM is that if measure figures do not exist, associated dimensions may fall out of the result. E.g., if we do not have posted any figures on a certain account, then no rows will be shown in reports that display accounts together with measures like actuals or budgeted amounts. Simply, the inner join between the facts and the ACCOUNT_D table, filters off such a records.

However, often we wish the ability to list all accounts, despite whether there are posted amounts associated. The ZEROLINE_D table solves this. It does so by providing a relationship between the core dimensions that are needed and which normally the fact tables establish relationship for: Company, Account, Currency and Fiscal Period. See Example 9.6 for how to utilize this feature. So in principle the table acts as a kind of proxy-fact table and a separate context is defined for it.

Account Groups

Account groups are **Reporting Structures** defined on the account and local account dimensions. From the ACCOUNT_D table, we join to the ACCOUNTGROUP_D which comprises REPORTINGSTRUCTUREHEADER and REPORTINGSTRUCTURELINE, and which offers various objects on these tables.

As the ACCOUNT_D is based on the performance view EXACCOUNTPV and this both has entries for accounts and local accounts, it is important to determine whether to run on local or global accounts. Therefore, the ACCOUNT_D table prompts for this and applies filtering of records depending on the answer from the user. See also Example 9.6 and Example 9.7 for how to utilize account grouping.

Retained Earnings

When reporting on financial balances (like we do with a Balance Sheet), often we want to include the year-end closing account. The account number of this account is stated in the system information of Maconomy. To the year-end result, we also need to add the profit and loss of the current year up to the given period; also known as **Retained Earnings**.

To facilitate the above, an alias YEARENDRESULTACCOUNT_D is made on the table ACCOUNT_D. The join is made in such a way, that we for all accounts get the year-end result account associated. This is the same account for all records fetched. Dedicated objects are created for this account such that we both can query balance accounts and the retained earnings and associate with these postings the year-end result account. For

the `YEARENDRESULTACCOUNT_D`, account groupings are also provided similar as for the `ACCOUNT_D`. See Example 9.7 for how to utilize this functionality in reports.

Chapter 10

Currency Exchange Universe

The Currency Exchange universe contains objects for performing currency conversion of amounts. By **Currency Conversion**, we understand converting an amount from one currency to another; e.g. the amount of 1000 USD to the corresponding amount in EUR. The universe is mainly used in combination with queries to the Finance universe where it is used in finance reports for making it possible to display the finance figures in a chosen reporting currency; given a certain exchange rate table and date. Hence, the purpose of the universe is to provide a currency conversion factor that can be multiplied on amounts in reports. The universe only contains dimension objects and attribute objects. It does not include measure objects as it is assumed that measures are fetched by means of queries to other universes.

10.1 Prerequisites

There are no prerequisites for using this universe. However, the universe offers the ability to apply **Company Specific Exchange Rates**, which needs to be set up for the companies. This is done by entering the name of an exchange rate table in the Miscellaneous field on the company card.

10.2 Maconomy Workflows

In the following sub-sections, we shall look at the common Maconomy workflows for setting up exchange rate tables and specifically for applying **Company Specific Exchange Rates** (). As these are not as such workflows about Maconomy transactions, we shall use the opportunity to explain how currency conversion and exchange tables work in Maconomy.

10.2.1 Setting up an exchange rate table

Each exchange rate table has a base currency for which each of the currencies in the table are calculated relatively to. E.g., an exchange rate table may have base in EUR which is stated on the table header. Each of the lines of the exchange rate table, represent a currency that we can convert amounts in EUR to; e.g. USD, DKK, etc. The base currency of the table does *not* occur in any of the lines. It is assumed that if we need to convert from the base currency to the base currency itself, the conversion factor is always 1.0. The header of the exchange rate table is stored in the database table `EXCHANGERATETABLE` and the lines in `EXCHANGERATE`.

To create a new exchange rate table, do the following:

1. In the window **Exchange Rate Tables**, create a new upper-pane record.
2. Enter the name "`Singapore`" and select "`EUR`" as base currency. Hit enter.
3. Create a number of lines in the table-panel. For each line, select a currency in the `Currency` field and enter an exchange rate in the field `Exchange Rate`. Note, that you can choose the same currency multiple times and assign different, non-overlapping date intervals.

When using the Currency Exchange universe for currency converning amounts, the universe will prompt the user for:

- a currency exchange table
- a conversion date
- a reporting currency (the currency to convert to)
- whether to use company specific exchange rates.

10.2.2 Company specific exchange rates

It is possible to assign a specific company specific exchange rate table to a company. If using currency specific exchange rates when reporting, instead of the currency exchange rate table that is prompted for, the exchange rate table set up on the company, is applied.

1. In the window **Company Information**, browse to the company in question.
2. In the field `Miscellaneous`, state the desired exchange rate table to be used as company specific exchange rate table.

In addition to the previously mentioned prompts, when using the Currency Exchange universe, reports offering currency conversion, also prompt the user for whether to use a reporting currency. In Table 10.1, we have stated the meaning of the combinations between two prompts: `Use Reporting Currency` and `Use Company Specific Exchange Rates` (referred to as *CSER*):

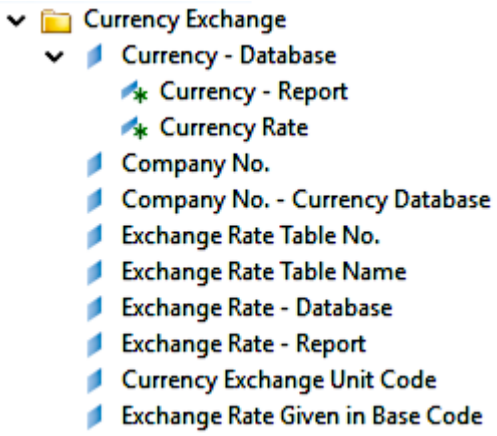
Use Reporting Currency	Use CSER	CSER is setup on company	Behaviour
No	No	No	No currency conversion
No	No	Yes	No currency conversion
No	Yes	No	No currency conversion
No	Yes	Yes	No currency conversion
Yes	No	No	Currency conversion using exchange rate table from prompt
Yes	No	Yes	Currency conversion using exchange rate table from prompt
Yes	Yes	No	Currency conversion using exchange rate table from prompt
Yes	Yes	Yes	Currency conversion using company's specific exchange rate table[1]

Table 10.1: Results of combination of prompt answers for `Use Reporting Currency` and `Use Company Specific Exchange Rates`

10.3 Business Layer

The universe provides a few dimensions and attributes for being able to currency convert amounts in measure objects fetched from other universes. Some objects themselves prompt for the above mentioned information (exchange rate table, date and whether to use **Company Specific Exchange Rates**) and provide list of values where needed.

The dimensions are rooted in the fields of the EXCHANGERATETABLE table and EXCHANGERATE. However, settings are also looked up in the SYSTEMINFORMATIONand for settings on **Company Specific Exchange Rates**, the COMPANYINFORMATION is used.
The object [Currency - Database] states the currency that we want to convert *from*. The object [Currency - Report] states the currency that we want to convert *to*. The object [Currency Rate] states the conversion factor that we need to apply to amounts in the *from-currency* in order to achieve the amount in the *to-currency*. The object [Company No. - Currency Database] states the company number concatenated with a currency. This object is used to bind the query of the Currency Exchange universe to the query of the universe providing the amounts. The other universe should contain a smilar object.



[Currency - Database]	EXCHANGERATE.CURRENCY where the associated exchange rate table is either the one stated by the user or the one set up on the company
[Currency - Report]	EXCHANGERATE.CURRENCY where the associated exchange rate table is either the one stated by the user or the one set up on the company
[Currency Rate]	The definition is complex and described below this table

10.4 Reporting Examples

10.4.1 Currency conversion of debit/credit amounts

Example 10.1 (Currency conversion of debit/credit amounts) *In this example, we shall create a report with two queries: (i) A simple query to the Finance universe for getting some debit and credit amounts that we want to currency convert, and (ii) a query to get the exchange rate for doing so.*

1. *Create a new WebIntelligence document and select the universe Finance as data*

source.

2. *Select the following objects for the query:*

   [Currency] *(folder [Company General Information])*

   [Company No. - Currency Database] *(folder [Company General Information])*

   [Company No.] *(folder [Company])*

   [Company Name] *(folder [Company])*

   [Account No.] *(folder [Account])*

   [Debit - Company] *(folder [Actuals - Company])*

   [Credit - Company] *(folder [Actuals - Company])*

3. *Add a query to the Currency Exchange universe.*

4. *Select the following objects for the query:*

   [Company No. - Currency Database]

   [Company No.]

   [Currency Database]

   [Currency Report] *(attribute of [Currency Database])*

   [Currency Rate] *(attribute of [Currency Database])*

5. *Add a filter on the [Company No. - Currency Database] object and select Result from other query using the object [Company No. - Currency]. This binds the two queries together and limits the currency exchange query to just the currencies available in the finance dataset.*

6. *Merge the two [Company No. Currency Database] objects*

7. *Merge the [Currency] object from the Finance universe with the [Currency Database] object from Currency Exchange universe.*

8. *Drag in the following objects to form a report table:*

   - *[Company No.]*
   - *[Account No.]*
   - *[Currency] (merged)*
   - *[Debit - Company]*
   - *[Credit - Company]*

9. *Edit the value cell of the two measures to multiply the measure with [`Currency Rate`]*

10.5 Data Foundation

The universe is centered around a single table named `CURRENCYEXCHANGE_D`. This table makes all of the data foundation. The table is complex as it in principle contains the algorithm of Maconomy's currency conversion[2]

CURRENCYEXCHANGE_D

The purpose of the table is—for each company—to provide a mapping from all the possible currencies to the selected reporting currency and the conversion rate between. The user will be prompted for:

- Whether to use Company Specific Exchange Rates
- Exchange rate table to use in case of not using Company Specific Exchange Rates
- Exchange rate date
- Reporting currency

In Table 10.3, we see a simplistic example of such a mapping:

Company No.	Currency	Reporting Currency	Conversion Factor
1	USD	EUR	0.92
1	EUR	EUR	1.0
1	DKK	EUR	0.13
2	USD	EUR	0.95

Table 10.3: Mapping from currency in the database to reporting currency and the conversion factor.

To understand the structure of the derived table, one first need to understand how Maconomy performs currency conversion. It is all about finding the conversion factor and we calculate that as the quotient between the rate of the from-currency and the rate of the to-currency. All exchange rate tables are defined to a have a certain currency as their base. The currencies in the table are then all the other currencies and for each we can lookup the rate between that currency and the base currency. If the currency has base in EUR and we want to get the rate for DKK, we have to lookup the DKK line in the table and take the rate between EUR and DKK. If it is the EUR currency we are looking for, obviously the rate is 1.0.

[2]However, the notion of job specific exchange rates is not supported.

When converting from one currency to another we thus first need to get the conversion rate for each of the currencies: The rate for the from-currency and the rate for the to-currency. The conversion factor is the quotient between these. If Maconomy is set up to have exchange rates in base, it is the from-rate divided by the to-rate. Else it is the other way around. So both for the from-rate and the to-rate, we need to make a select statement joining the exchange rate table with the lines.

For from-rate, we have:

```
SELECT
   COMPANYINFORMATION.COMPANYNUMBER,
   EXEXCHANGERATETABLE.EXCHANGERATETABLENUMBER,
   FROMEXCHANGERATE.CURRENCY,
   FROMEXCHANGERATE.EXCHANGERATE
FROM
   EXEXCHANGERATE FROMEXCHANGERATE
   INNER JOIN
      EXEXCHANGERATETABLE ON
         (FROMEXCHANGERATE.EXCHANGERATETABLENUMBER =
            EXEXCHANGERATETABLE.EXCHANGERATETABLENUMBER)
   INNER JOIN
      COMPANYINFORMATION ON
         (@Prompt('Use␣Company␣Specific␣Exchange␣Rates:',
                  'A',{'Yes','No'},mono,constrained,persistent,{'No'})='Yes' AND
         EXEXCHANGERATETABLE.EXCHANGERATETABLENUMBER =
            COMPANYINFORMATION.MISCELLANEOUSEXCHANGERATETAB)
         OR (@Prompt('Use␣Company␣Specific␣Exchange␣Rates:',
                     'A',{'Yes','No'},mono,constrained,persistent,{'No'})='No' AND
            EXEXCHANGERATETABLE.NAME = @Prompt(parExchangeRateTable))


WHERE
   FROMEXCHANGERATE.STARTINGDATE <= @Prompt(parExchangeRateDate)
   AND (FROMEXCHANGERATE.ENDINGDATE >= @Prompt(parExchangeRateDate)
         OR FROMEXCHANGERATE.ENDINGDATE IS NULL)

UNION ALL

SELECT
   COMPANYINFORMATION.COMPANYNUMBER,
   EXEXCHANGERATETABLE.EXCHANGERATETABLENUMBER,
   EXEXCHANGERATETABLE.REFERENCECURRENCY CURRENCY,
   (SELECT SYSTEMINFORMATION.CURRENCYEXCHANGEUNIT FROM SYSTEMINFORMATION) EXCHANGERATE
    FROM
      EXEXCHANGERATETABLE
      INNER JOIN
         COMPANYINFORMATION ON
            (@Prompt('Use␣Company␣Specific␣Exchange␣Rates:',
                     'A',{'Yes','No'},mono,constrained,persistent,{'No'})='Yes' AND
            EXEXCHANGERATETABLE.EXCHANGERATETABLENUMBER =
            COMPANYINFORMATION.MISCELLANEOUSEXCHANGERATETAB)
            OR (@Prompt('Use␣Company␣Specific␣Exchange␣Rates:',
                        'A',{'Yes','No'},mono,constrained,persistent,{'No'})='No' AND
               EXEXCHANGERATETABLE.NAME = @Prompt(parExchangeRateTable))
```

For to-rate, we have:

```
SELECT
   COMPANYINFORMATION.COMPANYNUMBER,
   EXEXCHANGERATETABLE.EXCHANGERATETABLENUMBER,
```

```
   TOEXCHANGERATE.CURRENCY,
   TOEXCHANGERATE.EXCHANGERATE
FROM
   EXEXCHANGERATE TOEXCHANGERATE
   INNER JOIN
   EXEXCHANGERATETABLE ON (TOEXCHANGERATE.EXCHANGERATETABLENUMBER =
      EXEXCHANGERATETABLE.EXCHANGERATETABLENUMBER)
INNER JOIN
   COMPANYINFORMATION ON
      (@Prompt('Use␣Company␣Specific␣Exchange␣Rates:',
                'A',{'Yes','No'},mono,constrained,persistent,{'No'})='Yes' AND
       EXEXCHANGERATETABLE.EXCHANGERATETABLENUMBER =
       COMPANYINFORMATION.MISCELLANEOUSEXCHANGERATETAB)
       OR (@Prompt('Use␣Company␣Specific␣Exchange␣Rates:',
                   'A',{'Yes','No'},mono,constrained,persistent,{'No'})='No' AND
           EXEXCHANGERATETABLE.NAME = @Prompt(parExchangeRateTable))
WHERE
   TOEXCHANGERATE.CURRENCY = @Prompt(parCurrency)
   AND TOEXCHANGERATE.STARTINGDATE <= @Prompt(parExchangeRateDate)
   AND (TOEXCHANGERATE.ENDINGDATE >= @Prompt(parExchangeRateDate)
        OR TOEXCHANGERATE.ENDINGDATE IS NULL)

UNION ALL

SELECT
   COMPANYINFORMATION.COMPANYNUMBER,
   EXEXCHANGERATETABLE.EXCHANGERATETABLENUMBER,
   EXEXCHANGERATETABLE.REFERENCECURRENCY CURRENCY,
   (SELECT SYSTEMINFORMATION.CURRENCYEXCHANGEUNIT FROM SYSTEMINFORMATION) EXCHANGERATE
    FROM
      EXEXCHANGERATETABLE
      INNER JOIN
         COMPANYINFORMATION ON
          (@Prompt('Use␣Company␣Specific␣Exchange␣Rates:',
                   'A',{'Yes','No'},mono,constrained,persistent,{'No'})='No' AND
          EXEXCHANGERATETABLE.EXCHANGERATETABLENUMBER =
          COMPANYINFORMATION.MISCELLANEOUSEXCHANGERATETAB)
          OR (@Prompt('Use␣Company␣Specific␣Exchange␣Rates:',
                      'A',{'Yes','No'},mono,constrained,persistent,{'No'})='Yes' AND
              EXEXCHANGERATETABLE.NAME = @PROMPT(parExchangeRateTable))
WHERE
   EXEXCHANGERATETABLE.REFERENCECURRENCY = @Prompt(parCurrency)
```

The difference between the two is really just that for the from-rate we need to represent all the currencies, whereas for the to-rate we want to limit to a single currency which is the one we prompt for. Now that we have the rate for both, we can combine on a higher level in the SQL:

```
   SELECT
      EXEXCHANGERATETABLE.EXCHANGERATETABLENUMBER,
      EXEXCHANGERATETABLE.NAME,
      EXEXCHANGERATETABLE.REFERENCECURRENCY,
      SYSTEMINFORMATION.CURRENCYEXCHANGEUNIT,
      SYSTEMINFORMATION.EXCHANGERATEGIVENINBASE,
      FROMEXCHANGERATE.COMPANYNUMBER FROMCOMPANYNUMBER,
      TOEXCHANGERATE.COMPANYNUMBER TOCOMPANYNUMBER,
      FROMEXCHANGERATE.CURRENCY FROMCURRENCY,
      TOEXCHANGERATE.CURRENCY TOCURRENCY,
      CASE WHEN FROMEXCHANGERATE.CURRENCY = EXEXCHANGERATETABLE.REFERENCECURRENCY
                  THEN SYSTEMINFORMATION.CURRENCYEXCHANGEUNIT
```



```
                ELSE FROMEXCHANGERATE.EXCHANGERATE END FROMRATE,
    CASE WHEN TOEXCHANGERATE.CURRENCY = EXEXCHANGERATETABLE.REFERENCECURRENCY
                THEN SYSTEMINFORMATION.CURRENCYEXCHANGEUNIT
                ELSE TOEXCHANGERATE.EXCHANGERATE END TORATE
  FROM
    EXEXCHANGERATETABLE
    INNER JOIN SYSTEMINFORMATION ON (1=1)
    INNER JOIN
    (
      -- here we have the select for the from-rate
    ) FROMEXCHANGERATE
    ON (FROMEXCHANGERATE.EXCHANGERATETABLENUMBER =
  EXEXCHANGERATETABLE.EXCHANGERATETABLENUMBER)
  INNER JOIN
    (
 -- here we have the select for the to-rate
    ) TOEXCHANGERATE ON (FROMEXCHANGERATE.EXCHANGERATETABLENUMBER =
 EXEXCHANGERATETABLE.EXCHANGERATETABLENUMBER)
) RATES
```

This all ends up in the following SQL:

```
SELECT
  SYSTEMINFORMATION.CURRENCYEXCHANGEUNIT,
  SYSTEMINFORMATION.EXCHANGERATEGIVENINBASE,
  RATES.FROMCOMPANYNUMBER,
  RATES.TOCOMPANYNUMBER,
  RATES.FROMCURRENCY,
  RATES.FROMRATE,
  RATES.TOCURRENCY,
  RATES.TORATE,
  RATES.EXCHANGERATETABLENUMBER,
  RATES.NAME,
  CASE WHEN RATES.FROMCURRENCY = RATES.TOCURRENCY THEN 1.0
       WHEN SYSTEMINFORMATION.EXCHANGERATEGIVENINBASE = 1
       THEN RATES.FROMRATE / RATES.TORATE
       ELSE RATES.TORATE / RATES.FROMRATE END CONVERSIONFACTOR
FROM
(
-- here we have the above comprised SQL for the from-rate and to-rate stated above
) WHERE
  RATES.FROMCOMPANYNUMBER = RATES.TOCOMPANYNUMBER
```

The reason we need the where clause in the bottom is because the two select statements for getting the from-rates and to-rates themselves involves the company because they have each of their own join to CompanyInformation. Obviously, all should be about one company so this is handled by this where clause.

The above algorithm for currency conversion is in fact a simplistic version of the one Maconomy performs. In Maconomy, so called job specific exchange rate tables can be used. This means that at the point where we lookup a currency in the exchange rate table, we may not get a result. Instead we look at the exchange rate tables reference to a standard table. Then we try looking it up in that standard table. If that does not succeed either, we continue with that standard table's standard table, and so forth. Obviously, that algorithm is not possible to express in SQL which is why we do not offer it in BPM. If a currency that we either want to convert from or to is not in the exchange rate table, customers need to add it in order for the finance reports to work.

Chapter 11

Bank Universe

The Bank universe provides objects for reporting on bank reconciliations and vendor payments[1]. Such payments could either be electronic payment or payments using checks. The universe also contains objects for reporting on the finance entries' debit and credit amounts. Thereby, it is possible to compare the reconciled and outstanding amounts with these actual G/L figures. For vendor payments, it is possible to distinguish the different kinds of payments as well as the different payment and reporting results; e.g. whether a check has been cashed or error reported.

The measure objects can be grouped and restricted by a variety of dimensions including company, account, vendor, dates and the ten standard dimensions.

11.1 Prerequisites

In order to fully benefit from the Bank universe, Maconomy should be set up such that:

- The system parameter **Use Calendar Day** is enabled and the calendar spans all necessary dates.

11.2 Maconomy Workflows

The purpose of the Bank universe is in some sense two-fold. On the one hand it provides the ability to report on bank reconciliations. On the other hand it provides the ability to report on electronic and manual payments of vendors.

[1] Notice, that there is also another universe—Vendor Payment—which is dedicated for reporting on payments done to vendors.

11.2.1 Bank Reconciliations

Bank reconciliations are in Maconomy represented by entries in the tables `BANKRECONCILIATIONHEADER` and `BANKRECONCILIATIONLINE`. Bank reconciliation is performed by matching postings against the transactions of a printed bank statement. In Maconomy an account can be associated with a bank account. When postings are done to that account, entries in the `BANKRECONCILIATIONLINE` are created for each `FINANCEENTRY`. Bank reconciliation is now the process of marking the `BANKRECONCILIATIONLINE` records which match on the printed statement from the bank. Before a bank reconciliation process has been started, there is always a record in the table `BANKRECONCILIATIONHEADER` for combination of **Payment Agent**, **Bank Registration Number** and **Bank Account Number**. This header represents the bank reconciliation lines that are *unreconciled*. The lines that can be reconciled are now marked and approved. In Maconomy, the header record gets a statement date, a number, and the reconciled lines are associated this header. The lines that were not reconciled are associated a new header that is automatically created. This new header represents the remaining unreconciled lines and does not yet have a statement date or a number.

Consider the following scenario:

1. In the window **External Accounts** locate one of the accounts with a payment agent related to cash.
2. Mark the field `Bank Reconciliation`.

   Postings that hit the account associated this external account, will now have the side effect that records in the `BANKRECONCILIATIONLINE` are created
3. Perform a few postings in the window **General Journal**. As offset, use the account that was associated the external account above.
4. In the window **Bank Reconciliations**, locate the combination of payment agent, bank registration number and bank account number, that designates the external account.
5. Mark some of the lines in the lower panel for reconciliation.
6. Approve the bank reconciliation.

   The reconciled lines will now be associated the approved bank reconciliation header. On the header we see that it represents the reconciled lines by the fact that it has a statement number. On the lines we see that they are reconciled by the fact that `BANKRECONCILIATIONLINE.RECONCILED` is true (`Yes`). The unreconciled lines are associated a new bank reconciliation header with a blank statement number. The lines associated that header all have `BANKRECONCILIATIONLINE.RECONCILED` stating false (`No.`)

11.2.2 Vendor Payments using Checks

Besides records in the `VENDORENTRY` table, vendor payments done electronically or by check, are represented as entries in the table `OUTPUTDATALINE`.

Consider the following scenario:

1. In the window **Purchase Orders** create a new purchase order and a line stating an invoice number and an amount of $5000.
2. Submit the purchase order.
3. In the window **Vendor Invoices** create a new journal.
4. In the lower panel enter the purchase order number, an invoice number, and the amount of the purchase order.
5. Set the payment mode to one of the *Check* types.
6. In the window **Invoice Allocation**, locate the vendor of the vendor invoice and submit for approval.
7. Bank in the **Vendor Invoices** window, post the vendor invoice journal.
8. In the window **Change Payment Selection by Vendor** locate the vendor of the vendor invoice.
9. Enter payment amount of $2000.
10. In the print window **Print Check**, set the payment mode to the *Check* type applied on the vendor invoice. Also enter a check number.
11. Print the check.

    A record has now been introduced in the table `OUTPUTDATALINE`. This record represents the fact that a payment of $2000 *has been issued. The record is associated a record in the `VENDORENTRY` which represents the vendor invoice.*
12. Back in the window **Change Payment Selection by Vendor** enter the rest of the amount to be paid: $3000.
13. In the print window **Print Check**, again set the payment mode to the *Check* type and print a new check.

    A new record has now been introduced in the table `OUTPUTDATALINE`; namely with an amount of $3000. *Both checks have the status* Issued *stated by the fact that the field `OUTPUTDATALINE.ISERRORREPORTED` or `OUTPUTDATALINE.REVERSED` states `Yes`.*

11.2.3 Error Reporting

When payments like checks have been issued, several things may happen. The payments could for various reasons be rejected or cancelled. Such status is received from the bank and registered in Maconomy. That registration will change the status of the output data lines.

11.3 Business Layer

The universe provides measures for reporting on amounts related to bank reconciliation and payments performed via the bank.

The business layer contains four folders with objects for the two kinds of measures:

Statements contains dimensions for reporting on the opening, closing, and movement of a bank reconciliation. The objects are defined as dimensions as it is not the idea that their value should be summarized across dimensions.

Reconciliations contains measures for reporting on the debit and credit values of reconciled entries, as well as associated dimensions.

Finance contains debit and credit amounts rooted in finance entries. These measures are used for matching the reconciled amounts against the actual finance entries to ensure that reconciliations are valid.

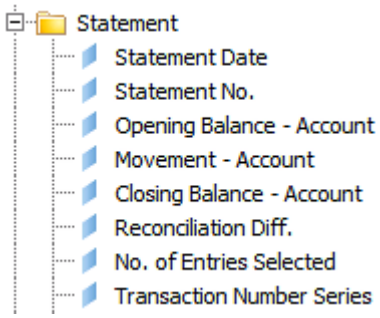
Paid – Registered – Discount contains measures for reporting on the paid, registered and cash discount amounts related to vendor payments.

The objects in these two folders are explained further below.

11.3.1 Statements

Statements are stored as records in the table `BANKRECONCILIATIONHEADER`. A *statement* represents a collection of finance entries that relate to bank statements; i.e. are related to bank transactions somehow. When doing a bank reconciliation, entries are marked to be selected in the statement. Each statement has an opening, a closing and a movement.

The folder contains the three dimension objects for reporting on opening, closing and movement of the statement. In addition, we have dimension objects for identifying statements, like the statement number, number of entries and transaction number series. Furthermore, we have a dimension object that states the remainder of the statement; i.e., the reconciliation difference.

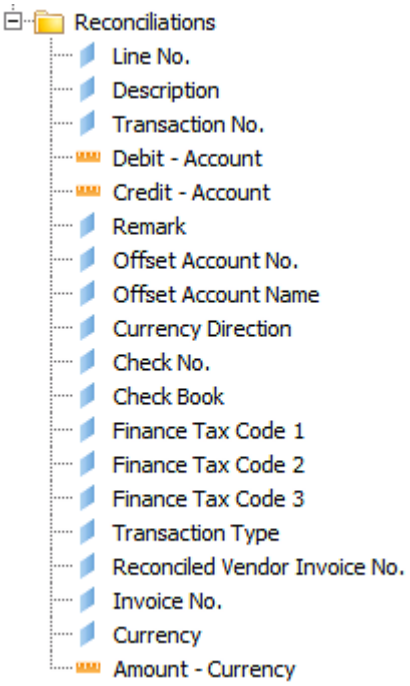


[Opening Balance - Account]	BANKRECONCILIATIONHEADER.STATEMENTOPENINGBALANCECURRE
[Movement - Account]	BANKRECONCILIATIONHEADER.STATEMENTMOVEMENTCURRENCY
[Closing Balance - Account]	BANKRECONCILIATIONHEADER.STATEMENTCLOSINGBALANCECURRE
[Reconciliation Diff.]	BANKRECONCILIATIONHEADER.RECONCILIATIONDIFFERENCE

11.3.2 Reconciliations

Reconciliations are stored as records in the table BANKRECONCILIATIONLINE. Basically, each finance entry which relates to a bank transaction has a corresponding record in BANKRECONCILIATIONLINE.

The folder contains three measures rooted in the BANKRECONCILIATIONLINE: [Debit - Account], [Credit - Account], and [Amount - Currency]. These measures can be used for reporting on the posted amounts that we wish to match against our written bank statement. Furthermore, the folder contains a number of dimensions for identifying individual lines, like the line number, description and transaction number. In addition, there are dimensions for offering more information about the lines, like the tax codes, vendor invoice number, invoiced number, check number and check book.

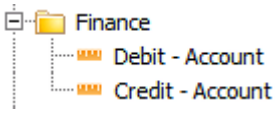


[Debit - Account]	BANKRECONCILIATIONLINE.DEBITCURRENCY
[Credit - Account]	BANKRECONCILIATIONLINE.CREDITCURRENCY
[Amount - Account]	BANKRECONCILIATIONLINE.AMOUNTCURRENCY

11.3.3 Finance

Finance entries are used for matching bank reconciliation entries in order to find out what remains unreconciled.

The folder contains just the two measure objects for reporting on debit and credit amounts of finance entries.

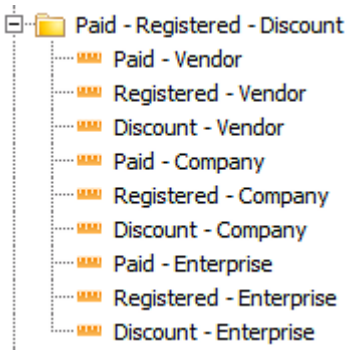


[Debit - Account]	FINANCEENTRY.DEBITCURRENCY
[Credit - Account]	FINANCEENTRY.CREDITCURRENCY

11.3.4 Vendor Payments via the Bank

Payments of vendors done via the bank, are in Maconomy stored as records in the table OUTPUTDATALINE. On such record, we can report on the paid amount, registered amount and discount (if any).

The folder contains the three measure objects for reporting on paid, registered and discount amounts; in the currency of the vendor, company and enterprise.



[Paid - Vendor]	OUTPUTDATALINE.PAYMENTCURRENCY
[Registered - Vendor]	OUTPUTDATALINE.REGISTEREDAMOUNTCURRENCY
[Discount - Vendor]	OUTPUTDATALINE.DISCOUNTCURRENCY
[Paid - Company]	OUTPUTDATALINE.PAYMENTBASE
[Registered - Company]	OUTPUTDATALINE.REGISTEREDAMOUNTBASE
[Discount - Company]	OUTPUTDATALINE.DISCOUNTBASE

[Paid - Enterprise] OUTPUTDATALINE.PAYMENTENTERPRISE
[Registered - Enterprise] OUTPUTDATALINE.REGISTEREDAMOUNTENTERPRISE
[Discount - Enterprise] OUTPUTDATALINE.DISCOUNTENTERPRISE

Chapter 12

Tax Settlement Universe

The Tax Settlement universe contains objects for reporting on tax settlements that have been done in Maconomy. It is possible to report on both **Basis Amounts** from which tax was calculated by Maconomy, and the individual **Tax Amounts** that was calculated. It is also possible to report on sub-categories of these amounts. For **Basis Amounts**, it is thus possible to distinguish basis amounts that are exempted from tax calculation and tax amounts that are typically not subject to payment of taxes, because they concern exporting. For tax amounts, it is possible to distinguish deferred and deductible/non-deductible tax amounts. A tax amount is considered deferred if it's tax code states that payment of taxes is not to be done until the corresponding invoice is paid. Deductible tax is the part of the tax amount which—according to the tax code—can be subtracted financially. The non-deductible tax is the tax amount that is not deductible.

The measures of the universe can be associated various dimensions that can be used for organizing various tax reports to legal authorities, collecting statistical tax information internally or externally, etc. Central dimensions are the company, the **Tax Reporting Unit**, the **Tax Type**, the **Tax Code**, and the **Tax Levels**.

When reporting on basis amounts, it is important to distinguish the amounts on the tax code or tax levels. The reason is that tax may be calculated for up to three levels. The basis amount on these levels may differ. A distinction on tax code or tax level hinders that the same basis amount is included multiple times when there are tax calculations for multiple levels.

When reporting on tax amounts, it is also possible to associate the tax code, tax type, and other attributes of the tax code. Furthermore, it is possible to report on the possible customers and vendors, or the customer or vendor invoice numbers, from which the tax originated.

In addition, it is possible to include the entry dates and tax dates in reporting on tax settlements. The entry date is the posting date of the tax entry in Maconomy. The tax

date is the date this tax entry is due. This determines which reporting period it was included in when the tax settlement was made in Maconomy.

12.1 Prerequisites

In order to fully benefit from the Tax Settlement universe, Maconomy should be set up such that:

- The system parameter **Use Calendar Day** is enabled and the calendar spans all necessary dates.
- Tax codes are set up correctly.
- Tax Settlement is utilized in Maconomy.[1]

12.2 Maconomy Workflows

When a company performs services or sells items, it is often values that are subject to taxes. The tax is added to the price; so if we sell hours for $800 and there is 25% in taxes, we invoice the customer $1000. The $800, we call the **Basis Amount**. The $200 we call the **Tax Amount**. In this case, we have a so-called **Tax Payable** because we need to pass on the $200 to the tax authorities. If we buy something, we pay taxes as part of paying the vendor invoice. If we buy a service of $1000 there may be $200 of these that are taxes. This tax amount we receive from tax authorities. It is thus **Tax Receivable**. So basically a tax settlement is about summarizing what we need to pay and what we need to receive and subtract these amounts. The result is what we in the end need to pay or receive. Most often, we need to pay more than we receive. However, for companies that export a lot, it may be the opposite.

Most postings in Maconomy give rise to the creation of entries that state the basis and tax amounts, as well as how these relate to the **Tax Reporting Unit**, to which they need to be reported. The entries are stored in the table `TAXSETTLEMENTENTRY`.

In the following sub-sections, we shall look at some common Maconomy workflows that are concerned with the data for which the Tax Settlement universe is made for reporting on. First, we shall look at a simple workflow that creates **Tax Payable**, then one for **Tax Receivable**, and finally a workflow with **Multiple Tax Levels**.

12.2.1 One level tax payable

In this workflow, we shall create a tax entry that is tax payable and includes it in a tax settlement:

[1] The Tax Settlement universe is primarily for reporting on data coming from tax settlements performed in Maconomy. However, it is possible to report on tax data by means of the universe, even though the tax settlement functionality in Maconomy is not used. The report query just needs another means for detecting the records in `VATSETTLEMENTENTRY` that are settled, like one of the date dimension objects.

1. In the window **Jobs**, create a new job on some customer. Use a company that only report taxes on one level.
2. In the window **Job Journal**, create a new journal.
3. Select an amount and fill out the other mandatory fields in the lower panel. Make sure that the amount is taxable.
4. Post the journal.
5. In the window **Invoice Selection**, locate the job.
6. Approve the selection and post the journal

The `VATSETTLEMENTENTRY` table will now contain an entry that states the basis amount, a tax amount on level one, the reporting unit, tax code, etc.

Consider also the `VATINFORMATION` table. This table holds entries for all the tax settlements that have been performed. It does so for each reporting unit. The settlements are enumerated for each period. Currently, the amounts may be zero for the present tax period, as we have not yet performed any tax settlements.

The entry can now be included in a tax settlement by the following workflow:

1. In the window **Tax Settlement**, locate the reporting unit of the entry.
2. Set the tax period in the upper panel (from/to month and year) if not already set
3. Create the settlement
4. Approve the settlement
5. Print the settlement

This workflow will only change the entry in the `TAXSETTLEMENTENTRY` table: It will associate a settlement number different from zero. This means that it belongs to the tax settlement represented by the record in the the `VATINFORMATION` table and having that settlement number.

In a real-life scenario, many entries may be listed as candidates to include in a tax settlement. For each entry, we may select whether to include it. The entries that are not selected for the tax settlement being done, will still have the settlement number zero. This means that they will appear in the list of candidates the next time a settlement is done for the reporting unit. That is, they are carried over to the next tax settlement that we will do.

12.2.2 One level tax receivable

In this workflow, we shall create a tax entry that is tax receivable and include it in a tax settlement:

1. In the window **Vendor Invoices**, create a new journal in the upper panel. Use a company that only reports taxes on one level.
2. Enter the vendor number, invoice number, amount, and other mandatory fields on a line in the lower panel.
3. Submit the journal.
4. Post the journal.

The `VATSETTLEMENTENTRY` now includes a new entry. Contrary to the previous entry we created, this entry has the settlement type *Tax Reveivable*.[2]

Including the entry in a tax settlement, is done in the same way as in the previous scenario.

12.2.3 Multiple tax levels

In the following workflow, we shall create multiple tax entries that relate to the same invoice. To do that, we need to make sure that the invoice (whether customer or vendor invoice) is created on a setup that requires calculation on multiple levels.

1. In the window **Jobs**, create a new job on some customer. Use a company that report taxes on two levels.
2. In the window **Job Journal**, create a new journal.
3. Select an amount and fill out the other mandatory fields in the lower panel. Make sure that the amount is taxable.
4. Post the journal.
5. In the window **Invoice Selection**, locate the job.
6. Approve the selection and post the journal

This should result in two new entries in `VATSETTLEMENTENTRY`: One for the tax amount on level 1, and one for a tax amount on level 2.

We see that for the two entries, we have the same basis amount but different levels and tax codes. Therefore, when reporting on the basis amounts, it is very important to include such dimensions, so that the basis amount of the two entries are not added together.

R Basis amounts should not be summarized across tax levels. Therefore, it is important to include measures like the tax level or tax code to distinguish them; else the result may be multiplication of these basis amount figures. When reporting on **Withholding Tax**, multiple lines may also be created stating the same basis amount. To deal with that, the dimension object [`Tax Account Dim. Comb. No.`] should be

[2]The tax settlement type is a popup so to see the meaningful display name, query `EXVATSETTLEMENTENTRY`.

used to distinguish the lines in the report. The dimension column could be defined as hidden in WebIntelligence as the dimension value typically does not add any business value to the report.

12.2.4 Tax exempt, export, deductible, and deferred

Basis amounts may be exempted from tax or related to export. Typically, this gives them a special status in the tax reporting because they are not subject to taxes.

The distinction on tax exempt and export, is defined by the **Tax Nature**. Usually the tax nature is `Taxable`, but **Tax Exempt** and **Export** are the two other options. These specify the two reasons for why the basis amount is not subject to tax.

The distinction on **Deductible Tax**/**Non-Deductible Tax** is defined on the tax code. A `RATIONUMBER` of e.g. 80 means that only 80% is deductible. Non-deductible is calculated as the remaining 20%.

Whether an amount is **Deferred Tax** id defined by the tax code. If the field `POSTVATASACCRUEDUNTILPAID` has the value `true`, the tax amount is deferred.

12.3 Business Layer

The universe provides measures for reporting on all sorts of tax amount, the amounts that have been the foundation for the tax amounts and associated dimensions related to the tax settlement process in Maconomy.

The business layer contains two folders for the two kinds of measures:

Basis Amounts containing objects displaying the foundation for the calculated tax amounts.

Tax Amounts containing objects displaying the tax amounts calculated by Maconomy.

The objects in these two folders are explained further below. However, before diving into these, we shall be concerned with the distinction between settled and unsettled taxes.

12.3.1 Settled and unsettled taxes

The Tax Settlement universe is able to distinguish settled and unsettled entries. When doing a tax settlement in Maconomy, tax entries that have not been settled are included and thereby marked so that they do not appear next time a settlement is done (also see earlier in this chapter).

In most tax reports, we wish to report on settled entries because it is assumed that we have done the tax settlement in Maconomy prior to running these reports. The dimension objects in the folder [`Tax Information`] are *bound* to the settled entries. Thus,

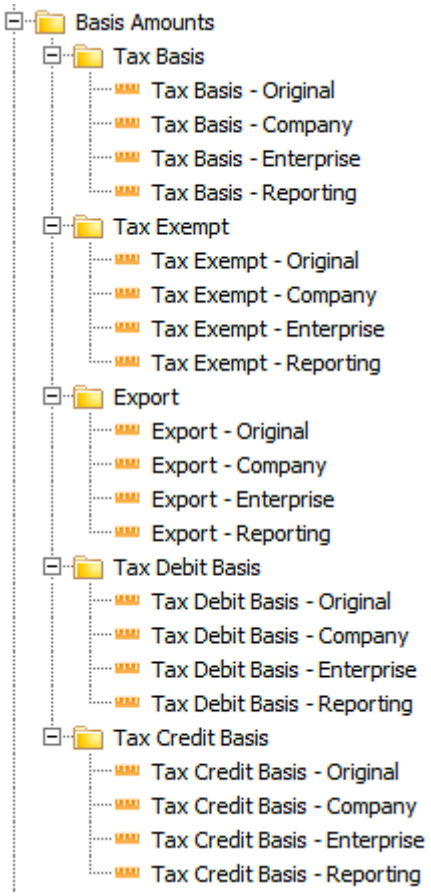
by prompting for a tax settlement number, displaying it or any other object from that folder, will ensure that only settled entries are included.

However, if we wish to report on all entries—settled or unsettled—we must avoid any object from the [`Tax Information`] folder.

12.3.2 Basis Amounts

Objects for reporting on basis amounts are located in the [`Basis Amounts`] folder. The basis amounts are typically useful when reporting statistics internally or to authorities. In principle it does not have a role to play in tax reporting as we here basically are interested in what tax to pay and what tax to receive. However, sometimes we want to include the basis amount in more general tax reports, or in order to add more transparency to the tax data.

There are three kinds of basis amounts; each available in four currency types: *Original*, *Company*, *Enterprise*, and *Reporting*. The `Tax Basis` objects display the full and unrestricted amounts from which the tax has been calculated. The `Tax Exempt` objects display the basis amounts for which tax is exempted. The `Export` objects display the basis amounts for which tax is not calculated because the amounts concern exporting. The folders [`Tax Debit Basis`] and [`Tax Credit Basis`] offer objects for reporting on the debit and credit sides of the tax basis amounts from the folder [`Tax Basis`].



[Tax Basis - Original] VATSETTLEMENTENTRY.DEBITBASISCURRENCY −VATSETTLEMENTENTRY.CREDITBASISCURRENCY
[Tax Basis - Company] VATSETTLEMENTENTRY.DEBITBASISBASE −VATSETTLEMENTENTRY.CREDITBASISBASE
[Tax Basis - Enterprise] VATSETTLEMENTENTRY.DEBITBASISENTERPRISE −VATSETTLEMENTENTRY.CREDITBASISENTERPRISE
[Tax Basis - Reporting] VATSETTLEMENTENTRY.DEBITBASISREPORTINGCURRENCY −VATSETTLEMENTENTRY.CREDITBASISREPORTINGCURRENCY
[Tax Basis - Original] VATSETTLEMENTENTRY.DEBITBASISCURRENCY −VATSETTLEMENTENTRY.CREDITBASISCURRENCY *if* EXVATSETTLEMENTENTRY.VATNATUREPN = 1 0otherwise
[Tax Basis - Company] VATSETTLEMENTENTRY.DEBITBASISBASE −VATSETTLEMENTENTRY.CREDITBASISBASE *if* EXVATSETTLEMENTENTRY.VATNATUREPN = 1 0otherwise
[Tax Basis - Enterprise] VATSETTLEMENTENTRY.DEBITBASISENTERPRISE −VATSETTLEMENTENTRY.CREDITBASISENTERPRISE *if* EXVATSETTLEMENTENTRY.VATNATUREPN = 1 0otherwise
[Tax Basis - Reporting] VATSETTLEMENTENTRY.DEBITBASISREPORTINGCURRENCY −VATSETTLEMENTENTRY.CREDITBASISREPORTINGCURRENCY *if* EXVATSETTLEMENTENTRY.VATNATUREPN = 1 0otherwise
[Export - Original] VATSETTLEMENTENTRY.DEBITBASISCURRENCY −VATSETTLEMENTENTRY.CREDITBASISCURRENCY *if* EXVATSETTLEMENTENTRY.VATNATUREPN = 2 0otherwise

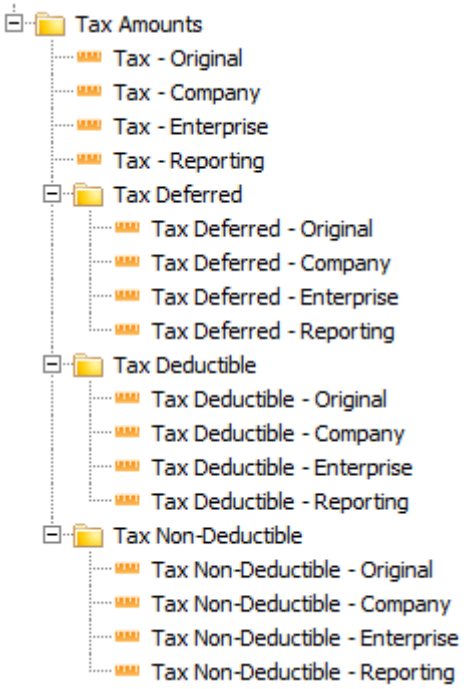
[Export - Company]	VATSETTLEMENTENTRY.DEBITBASISBASE −VATSETTLEMENTENTRY.CREDITBASISBASE *if* EXVATSETTLEMENTENTRY.VATNATUREPN = 2 0otherwise
[Export - Enterprise]	VATSETTLEMENTENTRY.DEBITBASISENTERPRISE −VATSETTLEMENTENTRY.CREDITBASISENTERPRISE *if* EXVATSETTLEMENTENTRY.VATNATUREPN = 2 0otherwise
[Export - Reporting]	VATSETTLEMENTENTRY.DEBITBASISREPORTINGCURRENCY −VATSETTLEMENTENTRY.CREDITBASISREPORTINGCURRENCY *if* EXVATSETTLEMENTENTRY.VATNATUREPN = 2 0otherwise
[Tax Debit Basis - Original]	VATSETTLEMENTENTRY.DEBITBASISCURRENCY
[Tax Debit Basis - Company]	VATSETTLEMENTENTRY.DEBITBASISBASE
[Tax Debit Basis - Enterprise]	VATSETTLEMENTENTRY.DEBITBASISENTERPRISE
[Tax Debit Basis - Reporting]	VATSETTLEMENTENTRY.DEBITBASISREPORTINGCURRENCY
[Tax Credit Basis - Original]	VATSETTLEMENTENTRY.CREDITBASISCURRENCY
[Tax Credit Basis - Company]	VATSETTLEMENTENTRY.CREDITBASISBASE
[Tax Credit Basis - Enterprise]	VATSETTLEMENTENTRY.CREDITBASISENTERPRISE
[Tax Credit Basis - Reporting]	VATSETTLEMENTENTRY.CREDITBASISREPORTINGCURRENCY

Basis amounts are the foundation from which Maconomy calculates the tax. However, the calculation is not always as simple as just applying the tax code percentage on the basis amount to get the tax amount. Many different principles may be applied when calculating tax; including **Tax on Tax** calculation. Therefore, one should be careful including the basis amounts just for proving the correctness of the calculated tax amounts.

12.3.3 Tax Amounts

Objects for reporting on tax amounts are located in the [`Tax Amounts`] folder.

There are four kinds of tax amounts; each available in four currency types: *Original*, *Company*, *Enterprise*, and *Reporting*. The `Tax` objects display the full and unrestricted tax amounts that have been calculated from the basis amounts. The `Tax Deferred` objects display the tax amounts which—according to the setup of the tax code—can be deferred. The `Tax Deductible` objects display the tax amounts or portion of such which—according to the setup of the tax code—can be deducted. The `Tax Non-Deductible` objects display the tax amounts or portion of such which—according to the setup of the tax code—cannot be deducted.



[Tax - Original]	(VATSETTLEMENTENTRY.DEBITVATCURRENCY −VATSETTLEMENTENTRY.CREDITVATCURRENCY) −(VATSETTLEMENTENTRY.DEBITVATNONDEDUCTIBLECURRENC −VATSETTLEMENTENTRY.CREDITVATNONDEDUCTIBLECURREN)
[Tax - Company]	(VATSETTLEMENTENTRY.DEBITVATBASE −VATSETTLEMENTENTRY.CREDITVATBASE) −(VATSETTLEMENTENTRY.DEBITVATNONDEDUCTIBLEBASE −VATSETTLEMENTENTRY.CREDITVATNONDEDUCTIBLEBASE)
[Tax - Enterprise]	(VATSETTLEMENTENTRY.DEBITVATENTERPRISE −VATSETTLEMENTENTRY.CREDITVATENTERPRISE) −(VATSETTLEMENTENTRY.DEBITVATNONDEDUCTIBLEENTERPR −VATSETTLEMENTENTRY.CREDITVATNONDEDUCTIBLEENTERP)

[Tax - Reporting]	(VATSETTLEMENTENTRY.DEBITVATREPORTINGCURRENCY −VATSETTLEMENTENTRY.CREDITVATREPORTINGCURRENCY) −(VATSETTLEMENTENTRY.DEBITVATNONDEDUCTIBLEREPORTI −VATSETTLEMENTENTRY.CREDITVATNONDEDUCTIBLEREPORT)
[Tax Deferred - Original]	VATSETTLEMENTENTRY.DEBITVATCURRENCY −VATSETTLEMENTENTRY.CREDITVATCURRENCY *if* EXVATCODE.POSTVATASACCRUEDUNTILPAIDPN = 1 0 otherwise
[Tax Deferred - Company]	VATSETTLEMENTENTRY.DEBITVATBASE −VATSETTLEMENTENTRY.CREDITVATBASE *if* EXVATCODE.POSTVATASACCRUEDUNTILPAIDPN = 1 0 otherwise
[Tax Deferred - Enterprise]	VATSETTLEMENTENTRY.DEBITVATENTERPRISE −VATSETTLEMENTENTRY.CREDITVATENTERPRISE *if* EXVATCODE.POSTVATASACCRUEDUNTILPAIDPN = 1 0otherwise
[Tax Deferred - Reporting]	VATSETTLEMENTENTRY.DEBITVATREPORTINGCURRENCY −VATSETTLEMENTENTRY.CREDITVATREPORTINGCURRENCY *if* EXVATCODE.POSTVATASACCRUEDUNTILPAIDPN = 1 0 otherwise
[Tax Deductible - Original]	(VATSETTLEMENTENTRY.DEBITVATCURRENCY −VATSETTLEMENTENTRY.CREDITVATCURRENCY)
[Tax Deductible - Company]	(VATSETTLEMENTENTRY.DEBITVATBASE −VATSETTLEMENTENTRY.CREDITVATBASE)
[Tax Deductible - Enterprise]	(VATSETTLEMENTENTRY.DEBITVATENTERPRISE −VATSETTLEMENTENTRY.CREDITVATENTERPRISE)
[Tax Deductible - Reporting]	(VATSETTLEMENTENTRY.DEBITVATREPORTINGCURRENCY −VATSETTLEMENTENTRY.CREDITVATREPORTINGCURRENCY)

[Tax Non-Deductible - Original] (VATSETTLEMENTENTRY.DEBITVATNONDEDUCTIBLECURRENC −VATSETTLEMENTENTRY.CREDITVATNONDEDUCTIBLECURREN)
[Tax Non-Deductible - Company] (VATSETTLEMENTENTRY.DEBITVATNONDEDUCTIBLEBASE −VATSETTLEMENTENTRY.CREDITVATNONDEDUCTIBLEBASE)
[Tax Non-Deductible - Enterprise] (VATSETTLEMENTENTRY.DEBITVATNONDEDUCTIBLEENTERPR −VATSETTLEMENTENTRY.CREDITVATNONDEDUCTIBLEENTERP)
[Tax Non-Deductible - Reporting] (VATSETTLEMENTENTRY.DEBITVATNONDEDUCTIBLEREPORTI −VATSETTLEMENTENTRY.CREDITVATNONDEDUCTIBLEREPORT)

For entries relating to a **Finance Tax Code**, the table EXFINANCEVATCODE is used instead.

12.4 Reporting Examples

12.4.1 Normal Tax reporting

In general, when reporting tax, the most important measure objects are simply the first tax objects in the [Tax Amounts] folder. The rest of the tax objects are just variants and can be used for giving a more detailed picture of the tax. But basically, the information needed in order to find out whether to pay or receive tax, are the first tax objects.

Often, we want to combine with dimensions displaying the tax settlement type (receivable, payable, etc.), and prompts for selecting which settlement number(s) to run the report for. These settlement numbers are in Maconomy succeeding integers; often following the number of the month but not necessarily. Therefore, the list of values of the object [Tax Settlement No.] (folder [Settlement] include the from/to year and month that this settlement spans time-wise.

In the following, we shall build a few tax reports starting with the most simple (but still useful) one.

Example 12.1 (A simple tax report) *In this example, we shall create the most simple (but valid) tax report possible.*

1. *Create a new WebIntelligence document and select the universe Tax Settlement as data provider.*
2. *Select the following objects for the query:*

   [Company No.] *(folder [Company])*

[Tax No.] *(folder [Company])*

[Reporting Currency] *(folder [Currency])*

[Tax Reporting Unit Name] *(folder [Tax Reporting Unit])*

[Tax Settlement No.] *(folder [Settlement])*

[Tax Settlement Type] *(folder [Settlement])*

[Tax - Reporting] *(folder [Tax Amounts])*

3. *Drag the object [Tax Settlement No.] down as a filter and make it a mandatory prompt. Use the* `equal` *condition to make it a single value prompt.*
4. *Create a report with a table containing all the objects except [Tax Settlement No.] and [Company No.].*
5. *Drag the object [Tax No.] up to create a section. Edit the section cell so it appends the value of [Company No.] with some spaces between.*
6. *Drag the object [Reporting Currency] up to create a nested section.*
7. *Drag the object [Tax Reporting Unit Name] up to create yet another nested section. The sections (out to in) should thus be: [Tax No.], [Reporting Currency], and [Tax Reporting Unit Name].*
8. *In the cells displaying [Tax - Reporting], edit the formula to display the absolute value.*
9. *Add a total with the title "Tax to Pay:", and summarize the values of the object [Tax - Reporting];* not *the absolute value of it. Display the value negated.*
10. *Refresh the document*

We now have a very simple report that displays what to pay in taxes for a given period (or more precisely for the settlement number selected when running the report). Tax amounts for ***Tax Receivable*** *are in the Maconomy database positive; where as for* ***Tax Payable***, ***Investment Tax*** *and others, is negative. So it can be more natural—now that we also display the type—to display the absolute values.*

However, when we display how much is for paying, we need the tax values with sign. In this example, we focused on what to pay, so we had to negate the summarized value. The value is represented with a sign that is in the company's favor. The sign (direction) of the tax can also be reported on by including the object [Tax Sign]. ■

Example 12.2 (Tax report with tax codes and tax levels) *In this example, we elaborate on the simple tax report from the previous example.*

1. *Save the report from Example 12.1 with a new document name.*

2. *Edit the query to also include the following objects:*

   [Tax Code Name] *(folder [Tax Code])*

   [Tax Level Name] *(folder [Settlement])*

3. *Add the two new objects as additional columns (left to the tax amount column).*

4. *Refresh the document.*

We now see that data are displayed in more detail. Where we earlier just had one line per settlement type, we may now have multiple lines. If taxes have been calculated on multiple levels, we see multiple lines; one for each level.

For each level, we also see which tax code that was applied on the different levels. This could mean that we get even more lines, because we could have applied different tax codes on different entries all on level 1; and similar for the other tax levels. ■

12.4.2 Reporting on unsettled entries

The Tax Settlement universe also provides the ability to report on which entries that are not settled yet. As explained earlier, tax entries (records in the table `VATSETTLEMENTENTRY` are "born" with the value zero in the field `VATSETTLEMENTNUMBER`. When the entry is included in a tax settlement in Maconomy, that value is assigned the value of the settlement (larger than zero). This way, we can report on entries that are possible to select for a tax settlement process, group and slice/dice them any way we like to provide the information necessary to take decisions in various tax settlement and reporting situations.

R Reports on unsettled entries must not use objects from the folder [`Tax Information`] as these will tie the fact entries to tax settlements made in Maconomy. As unsettled entries have not yet been associated with a settlement (thereby *unsettled*) they would be filtered off.

Example 12.3 (Unsettled entries report) *In this example, we shall create a simple tax report which lists the entries that are not settled yet.*

1. *Create a new WebIntelligence document and select the universe Tax Settlement as data provider.*

2. *Select the following objects for the query:*

   [Company No.] *(folder [Company])*

   [Tax No.] *(folder [Company])*

   [Reporting Currency] *(folder [Currency])*

   [Tax Reporting Unit Name] *(folder [Tax Reporting Unit])*

[**Tax Settlement Type**] *(folder [Settlement])*

[**Tax Basis - Reporting**] *(folder [Basis Amounts])*

[**Tax Code Name**] *(folder [Tax Code])*

[**Tax - Reporting**] *(folder [Basis Amounts])*

3. *Add a filter on the object [Settled Code] (folder [Settlement]) and require it equal to zero.*
4. *Add a filter on the object [Tax Date] using the* **between** *operation so that we offer a from- and an end-date prompt.*
5. *Create a report with a table similar as done in the previous examples*
6. *Refresh the document*

This report will list the amounts to be settled, grouped by settlement type and tax code. The user is prompted for selecting a date interval for restricting the entries on the tax date. This is the date the entries are due. More dimensions that break the lines of the table into more detail, could be added for making the report more useful. ■

12.5 Data Foundation

The universe is centered on a single fact table named `VATSETTLEMENTENTRY_D` which is based on `VATSETTLEMENTENTRY`.

VATSETTLEMENT_D

The table is composed by selects to the following database tables/views:

`BOVATSETTLEMENTENTRY` the BO view on `VATSETTLEMENTENTRY` which provides all the fact figures and dimension values for joining to dimension tables; all except for the value making it possible to join to the tax code tables.

`JOURNAL` the database table which is used for determining whether to associate a tax code from the `VATCODE` table or a tax code from the `FINANCEVATCODE` table.

The fact table is joined to dimension tables that offer additional information related to tax figures:

`SYSTEMINFORMATION_D` which provides the ability of reporting on the enterprise currency.

`COMPANYINFORMATION_D` which provides the ability of reporting on company number, name and tax number.

`EXVATREPORTINGUNIT` which provides the ability of reporting on the reporting unit and information associated.

`CUSTOMER_D` which provides the ability of reporting on associated customers and their contact information.

`VENDOR_D` which provides the ability of reporting on associated vendors and their contact information.

`CUSTOMERENTRY_D` which provides a link to `INVOICE_D` so it is possible to report on customer invoice numbers and date.

`VENDORENTRY_D` which provides a link to `VENDORINVOICEJOURNAL_D` so it is possible to report on vendor invoice numbers and date.

`VATINFORMATION_D` which provides the ability of reporting on date of settlements done in Maconomy; including the settlement numbers.

`VATDATE` which is an alias of `CALENDARDAYPV` and provides the ability of reporting the date tax entries are due.

`ENTRYDATE` which is an alias of `CALENDARDAYPV` and provides the ability of reporting the date tax entries are created.

`VATCODE_D` which provides the ability of reporting on the tax codes of the tax entries.

VATCODE_D

Some tax entries associate to a **Finance Tax Code** (table `FINANCEVATCODE` where others associate to a **Tax Code** (table `VATCODE`). From the journal type of a tax entry, we can determine whether to join to the one or the other table. This is why we need to join `VATSETTLEMENTENTRY` with `JOURNAL` in the fact table of the universe.

The dimension table `VATCODE_D` is defined as a union between the table `FINANCEVATCODE` and `VATCODE`. Each part of the union defines (among other fields) the field `CODETYPE` with the following values:

FINANCEVATCODE for the `FINANCEVATCODE` union part.

VATCODE for the `VATCODE` union part.

In the fact table, we select a field `CODETYPE` that has values depending on the journal type. The join can thereby be defined on the tax code names and the code type.

Chapter 13

GL Audit Universe

The GL Audit universe contains objects for reporting on financial figures and subledger figures possibly related to tax data. The universe is specielt in the sense that it only contains dimension objects and thus do not perform any form for aggregation. The purpose of the universe is to serve as foundation for doing **Data Exports** that list finance entries, customer entries and vendor entries possibly associated data from vat settlement entries. Hence, the universe is not suitable as data source in normal WebI reports. To finance entry data, it is possible to associate jobs, activities, vendor invoice information, item information, company information and journal information. To the customer and vendor entry data, it is possible to associate company information. For all sorts of entries, it is possible restrict on fiscal year and dates. It is also possible to restrict possibly associated tax settlement entries on their entry date. Furthermore, it is possible to report on set up company information like the fiscal year setup of companies in order to get the start and end date of a given fiscal year. For some dimensions, it is possible to get those used in transactional records or those independent of transactions (i.e. the setup of dimensions).

13.1 Prerequisites

No prerequisites are needed for utilizing the universe.

13.1.1 Special universe properties

The GL Audit universe is specially designed for data export. Therefore it possesses some special properties that we will explain below.

Finance and subledger data combined with tax data

The universe offers the ability to do data exports of finance and sub-ledger data combined with tax data. It does so by offering the ability to query on records from `FINANCEENTRY`,

CUSTOMERENTRY and VENDORENTRY — each combined with possibly associated records from VATSETTLEMENTENTRY. The universe is organised more technically around the mentioned tables. This means that it has fact tables based on each of the three tables FINANCEENTRY, CUSTOMERENTRY and VENDORENTRY, and in each fact table the table in question is outer-joined with VATSETTLEMENTENTRY. For some records, there is a contribution from the VATSETTLEMENTENTRY, whereas for others there is not. For some records there may even be multiple records in VATSETTLEMENTENTRY matching. This results in that the (e.g.) finance entry record's data is repeated on the multiple records in the query result. Hence, processing the query result in the right way is highly important. The same goes for customer entries and vendor entries. If no tax data is associated, the objects defined on the VATSETTLEMENTENTRY table are null.

Flat structure without aggregation

Because the universe is aimed for data exports where it is crucial to fetch the individual data records (e.g. the individual finance entries), the universe does not have any measures. All objects are dimension objects. Thus, no aggregation is applied and the data set retrieved from queries is merely a flat structure of data. Any summation or other aggregation must be done in the extension code of the data export querying the universe. As an example, consider three vendor invoices leading to the creation of 6 different finance entries. Making a query that includes the debit and credit amounts of the finance entries, along with dimensions that are the same for all six entries (e.g. vendor number), will result in six records. This is completely different from when querying the Finance universe where record would be comprised and amounts summarized.

Handling multiple tax levels

If a transaction involves calculation of tax on multiple levels, multiple records in VATSETTLEMENTENTRY will be associated the finance entry, customer entry or vendor entry in question. Because no data are aggregated, the records retrieved will repeat the data from the finance, customer or vendor entry for as many times as there are vat settlement entries associated. As mentioned earlier, it is up to the Java code of the data export extension to hanlde any possible redundancy of data.

13.2 Maconomy Workflows

In the following sub-sections, we shall look at the common Maconomy workflows that are concerned with the data for which the GL Audit universe is made for reporting on. We shall concentrate on the special relationship between finance entries and vat settlement entries. This relationship characterises the properties of the universe of being without measures and without aggregation. The records then need to be processed in a special way as we will explain below.

13.2.1 Finance transactions with tax settlement entries associated

Finance transactions are made as also described in Section 9.2, but in fact many workflows in Maconomy lead to the creation of finance entries. The scenario below is just one of many simple examples in which finance entries are created along with vat settlement entries. The direct way of creating finance entries is to use the **General Journal** where we can actively assign tax codes in two levels. This gives us two vat settlement entries associated the same finance entry which demonstrates the special characteristics of the universe.

1. In the window **Company Information** create a new company with company number `"51"`.
2. Setup the company to have tax on two levels. We shall assume that we have two tax codes available: Tax Level 1 for level 1 and Tax Level 2 for level 2[1]
3. In the window **General Journal**, create a new journal on company `"51"`.
4. Create a line of type `G`, and enter a credit amount using offset `Cash`, and stating `Tax Level 1` for level 1 tax and `Tax Level 2` for level 2 tax.
5. Post the journal

The result is the creation of (depending on setup) typically 4 finance entries. Two fo these are associated separate records in the table `VATSETTLEMENTENTRY` via the field `BASISFINANCEENTRYKEY` that points to the finance entries' instance key.

When reporting on data like this (typically by means of **Data Exports**, the same finance entry record is displayed twice; once for each of the vat settlement entry records. This means that the processing of the data needs to avoid counting the finance entry data twice. E.g. the data from the first of the entries could be processed, then the vat settlement entry for the first record could be processes followed by the vat settlement entry of the second record. This way we process the finance entry data once and the vat settlement entry data twice; one time for each of the two vat settlement entry records. Similar scenarios can be done creating entries in the tables `CUSTOMERENTRY` and `VENDORENTRY`, and the records in these tables can also be associated records in `VATSETTLEMENTENTRY`; also on multiple levels. The principle for processing the data fetched from the universe, is the same.

13.3 Business Layer

The universe provides dimension objects for reporting on finance, customer and vendor entries and data from possibly associated vat settlement entries. All objects (also those for reporting on amounts) are dimensions for reasons explained earlier in this chapter.

[1] This can be set up in various ways. It does not matter which way you choose as long as we get tax settled on two levels.

The business layer contains a main folder for each of these three sorts of data:

Finance Tax Entry containg objects displaying finance entry data and possibly vat settlement entry data associated.

Customer Tax Entry containg objects displaying customer entry data and possibly vat settlement entry data associated.

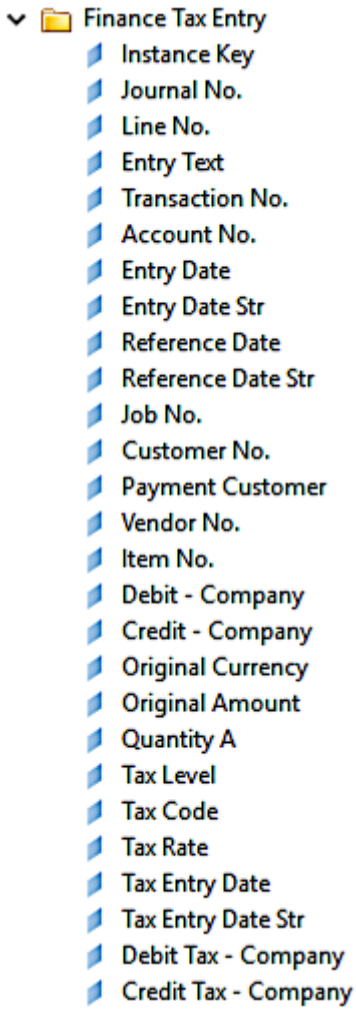
Vendor Tax Entry containg objects displaying vendor entry data and possibly vat settlement entry data associated.

In the following sub-sections, we shall go through each of these:

13.3.1 Finance Tax Entry

Objects for querying finance entry data and associated vat settlement entry data, are located in the [`Finance Tax Entry`] folder. The folder contains objects for querying dimensions and central amount fields from the finance entry, as well as similar for vat settlement entries associated the finance entry.

The first objects in the folder are objects defined on `Finance Entry`, whereas the last objects are defined on `VATSETTLEMENTENTRY`. The object [`Instance Key`] is an important field—not for reporting—but for iterating through entries in the result set in a grouped way when multiple vat entries are related to the same finance entry.

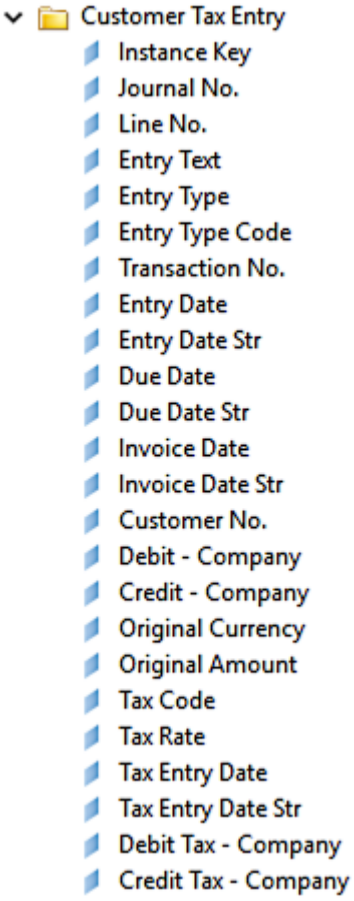


[Instance Key]	EXFINANCEENTRY.INSTANCEKEY
[Journal No.]	EXFINANCEENTRY.JOURNALNUMBER
[Line No.]	EXFINANCEENTRY.LINENUMBER
[Entry Text]	EXFINANCEENTRY.ENTRYTEXT
[Transaction No.]	EXFINANCEENTRY.TRANSACTIONNUMBER
[Account No.]	EXFINANCEENTRY.ACCOUNTNUMBER

[Entry Date]	EXFINANCEENTRY.ENTRYDATE
[Entry Date Str]	EXFINANCEENTRY.ENTRYDATEDS
[Reference Date]	EXFINANCEENTRY.REFERENCEDATE
[Reference Date Str]	EXFINANCEENTRY.REFERENCEDATEDS
[Job No.]	EXFINANCEENTRY.JOBNUMBER
[Customer No.]	EXFINANCEENTRY.CUSTOMERNUMBER
[Payment Customer]	EXFINANCEENTRY.PAYMENTCUSTOMER
[Vendor No.]	EXFINANCEENTRY.VENDORNUMBER
[Item No.]	EXFINANCEENTRY.ITEMNUMBER
[Debit - Company]	EXFINANCEENTRY.DEBITBASE
[Credit - Company]	EXFINANCEENTRY.CREDITBASE
[Original Currency]	EXFINANCEENTRY.ORIGINALCURRENCY
[Original Amount]	EXFINANCEENTRY.ORIGINALAMOUNT
[Quantity A]	EXFINANCEENTRY.QUANTITYA
[Tax Level]	EXVATSETTLEMENTENTRY.VATLEVEL
[Tax Code]	EXVATSETTLEMENTENTRY.VATCODE
[Tax Rate]	EXVATSETTLEMENTENTRY.VATRATE
[Tax Entry Date]	EXVATSETTLEMENTENTRY.ENTRYDATE
[Tax Entry Date Str]	EXVATSETTLEMENTENTRY.ENTRYDATEDS
[Debit Tax - Company]	EXVATSETTLEMENTENTRY.DEBITVATBASE
[Credit Tax - Company]	EXVATSETTLEMENTENTRY.CREDITVATBASE

13.3.2 Customer Tax Entry

Objects for querying customer entry data and associated vat settlement entry data, are located in the [`Customer Tax Entry`] folder. The folder contains objects for querying dimensions and central amount fields from the customer entry, as well as similar for vat settlement entries associated the customer entry.



The first objects in the folder are objects defined on `Customer Entry`, whereas the last objects are defined on `VATSETTLEMENTENTRY`. The object [`Instance Key`] is an important field—not for reporting—but for iterating through entries in the result set in a grouped way when multiple vat entries are related to the same customer entry.

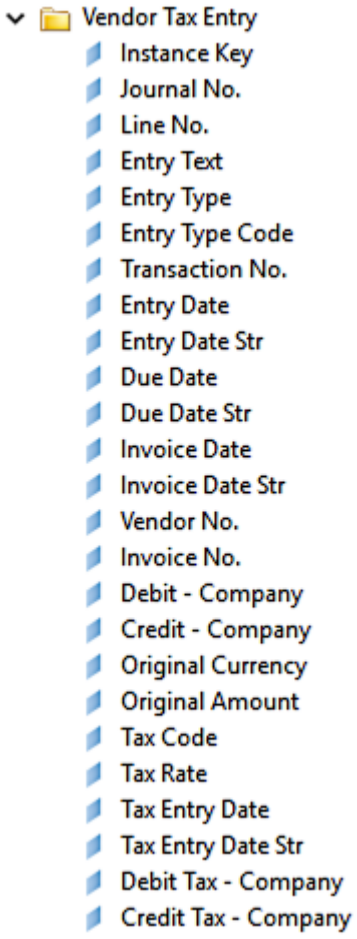
[`Instance Key`]	`EXCUSTOMERENTRY.INSTANCEKEY`
[`Journal No.`]	`EXCUSTOMERENTRY.JOURNALNUMBER`
[`Line No.`]	`EXCUSTOMERENTRY.LINENUMBER`

[Entry Text]	EXCUSTOMERENTRY.ENTRYTEXT
[Entry Type]	EXCUSTOMERENTRY.ENTRYTYPE
[Entry Type Code]	EXCUSTOMERENTRY.ENTRYTYPEPN
[Transaction No.]	EXCUSTOMERENTRY.TRANSACTIONNUMBER
[Account No.]	EXCUSTOMERENTRY.ACCOUNTNUMBER
[Entry Date]	EXCUSTOMERENTRY.ENTRYDATE
[Entry Date Str]	EXCUSTOMERENTRY.ENTRYDATEDS
[Due Date]	EXCUSTOMERENTRY.DUEDATE
[Due Date Str]	EXCUSTOMERENTRY.DUEDATEDS
[Invoice Date]	EXCUSTOMERENTRY.INVOICEDATE
[Invoice Date Str]	EXCUSTOMERENTRY.INVOICEDATEDS
[Customer No.]	EXCUSTOMERENTRY.CUSTOMERNUMBER
[Debit - Company]	EXCUSTOMERENTRY.DEBITBASE
[Credit - Company]	EXCUSTOMERENTRY.CREDITBASE
[Original Currency]	EXCUSTOMERENTRY.ORIGINALCURRENCY
[Original Amount]	EXCUSTOMERENTRY.ORIGINALAMOUNT
[Quantity A]	EXCUSTOMERENTRY.QUANTITYA
[Tax Code]	EXVATSETTLEMENTENTRY.VATCODE
[Tax Rate]	EXVATSETTLEMENTENTRY.VATRATE
[Tax Entry Date]	EXVATSETTLEMENTENTRY.ENTRYDATE
[Tax Entry Date Str]	EXVATSETTLEMENTENTRY.ENTRYDATEDS

[Debit Tax - Company]	EXVATSETTLEMENTENTRY.DEBITVATBASE
[Credit Tax - Company]	EXVATSETTLEMENTENTRY.CREDITVATBASE

13.3.3 Vendor Tax Entry

Objects for querying vendor entry data and associated vat settlement entry data, are located in the [Vendor Tax Entry] folder. The folder contains objects for querying dimensions and central amount fields from the vendor entry, as well as similar for vat settlement entries associated the finance entry.



The first objects in the folder are objects defined on Vendor Entry, whereas the last objects are defined on VATSETTLEMENTENTRY. The object [Instance Key] is an important field—not for reporting—but for iterating through entries in the result set in a grouped way when multiple vat entries are related to the same vendor entry.

[Instance Key]	EXVENDORENTRY.INSTANCEKEY

[Journal No.]	EXVENDORENTRY.JOURNALNUMBER
[Line No.]	EXVENDORENTRY.LINENUMBER
[Entry Text]	EXVENDORENTRY.ENTRYTEXT
[Entry Type]	EXVENDORENTRY.ENTRYTYPE
[Entry Type Code]	EXVENDORENTRY.ENTRYTYPEPN
[Transaction No.]	EXVENDORENTRY.TRANSACTIONNUMBER
[Account No.]	EXVENDORENTRY.ACCOUNTNUMBER
[Entry Date]	EXVENDORENTRY.ENTRYDATE
[Entry Date Str]	EXVENDORENTRY.ENTRYDATEDS
[Due Date]	EXVENDORENTRY.DUEDATE
[Due Date Str]	EXVENDORENTRY.DUEDATEDS
[Invoice Date]	EXVENDORENTRY.INVOICEDATE
[Invoice Date Str]	EXVENDORENTRY.INVOICEDATEDS
[Customer No.]	EXVENDORENTRY.VENDORNUMBER
[Debit - Company]	EXVENDORENTRY.DEBITBASE
[Credit - Company]	EXVENDORENTRY.CREDITBASE
[Original Currency]	EXVENDORENTRY.ORIGINALCURRENCY
[Original Amount]	EXVENDORENTRY.ORIGINALAMOUNT
[Quantity A]	EXVENDORENTRY.QUANTITYA
[Tax Code]	EXVATSETTLEMENTENTRY.VATCODE
[Tax Rate]	EXVATSETTLEMENTENTRY.VATRATE

[Tax Entry Date]	EXVATSETTLEMENTENTRY.ENTRYDATE
[Tax Entry Date Str]	EXVATSETTLEMENTENTRY.ENTRYDATEDS
[Debit Tax - Company]	EXVATSETTLEMENTENTRY.DEBITVATBASE
[Credit Tax - Company]	EXVATSETTLEMENTENTRY.CREDITVATBASE

Chapter 14

AR Aging Universe

The AR Aging universe provides objects for reporting on outstanding customer balances. The balances are the amounts that customers have not paid concerning invoices and which have not been reconciled yet. The outstanding balances can be aged according to the aging principles set up in Maconomy and thereby distinguish the outstandings on how old they are. The universe also provides objects for reporting on customer invoices as well as paid amounts. Furthermore, it is possible to report on the additional types of entries like cash discounts and exchange rate gains and losses. It is possible to report on whether entries are blocked and the reason they may be blocked.

The measure objects can be grouped and restricted by a variety of dimensions including company, customer, job, employee, dates, and the ten standard dimensions.

14.1 Prerequisites

In order to fully benefit from the AR Aging universe, Maconomy should be set up such that:

- The system parameter **Use Calendar Day** is enabled and the calendar spans all necessary dates.
- In order to report on aging figures, an aging principle needs to be set up.

14.2 Maconomy Workflow

In the following sub-sections we shall look at the common Maconomy workflows that are concerned with the data for which the AR Aging is made for reporting on. First, we shall look at a classical invoice-payment-reconciliation workflow. Then we shall consider variants of that flow, including partial reconciliations, creation of cash discounts and blocked entries.

14.2.1 Customer Invoices, Payments, and Reconciliations

Customer invoices and payments are represented by records in various tables. Behind any kind of invoice (except invoices on account) there is a record in the the `CUSTOMERENTRY` table. Similarly, customer payments are represented by records in the `CUSTOMERENTRY` table. Invoices and payments are matched in order to state which payments relate to which invoices. E.g., one payment could cover two or more invoices, or a payment may cover only part of the amount of an invoice. The process of matching customer entries representing invoices with corresponding entries matching payments, is called **Customer Reconciliations** and is done in the window **Customer Open Entry Reconciliation**.

When a reconciliation has been approved, records of two additional tables, are created. A record in the table `CUSTOMERRECONCILIATIONJOURNAL` expresses what has been reconciled against the record in the upper panel of the window **Customer Open Entry Reconciliation**. One or more records `CUSTOMERRECONCILIATION` express what has been reconciled against each customer entry (invoices or credit memos) in the lower panel.

Reconciliations can also be removed. When this is done an additional record is created in the table `CUSTOMERRECONCILIATIONJOURNAL`. This record "points" to the existing record in the same table and states that it *removes* that reconciliation.

Consider the following:

1. In the window **Jobs**, we create a new invoiceable job on a customer.
2. In the window **Job Journal**, we create a new journal and enter the job, an invoiceable activity, and a quantity of 100.
3. We invoice the amount as it is in the **Invoice Selection** window.

   This provides a `CUSTOMERENTRY` record.

4. In the window **Customer Payments**, create a payment on the same amount as the invoiced amount.

   This provides an additional `CUSTOMERENTRY` record.

5. In the window **Customer Open Entry Reconciliation**, reconcile the payment and the invoice fully.

   This closes the two customer entries and creates a record in the `CUSTOMERRECONCILIATION` table and a record in the `CUSTOMERRECONCILIATIONJOURNAL` table.

14.2.2 Fully and Partial Reconciliations

Customer reconciliations can be fully or partial. If a reconciliation is done fully, all the involved entries are closed. If it is done partially, some entries will be left open and need to be involved in additional reconciliations to be closed. An example is if an invoice is paid in two rates; i.e., by two payments.

Consider a scenario similar to the above:

1. In the window **Jobs** we create a new invoiceable job on customer *10030*
2. In the window **Job Journal** we create a new journal and enter the job, an invoiceable activity and a quantity of 250.
3. We invoice the amount as it is in the **Invoice Selection** window.
4. In the window **Customer Payments**, create a payment on half the invoice amount.
5. In the window **Customer Open Entry Reconciliation** enter just half of the amount to reconcile and choose to reconcile partially.

   This closes the payment customer entry and leaves the invoice customer entry open.

14.2.3 Customer Payments and Cash Discounts

Cash discounts and other kinds of deviations may be created or occur when reconciling customer entries.

Again, consider the scenario from before:

1. In the window **Jobs**, we create a new invoiceable job on a customer.
2. In the window **Job Journal**, we create a new journal and enter the job, an invoiceable activity and a quantity of 200.
3. We invoice the amount as it is in the **Invoice Selection** window.
4. In the window **Customer Payments**, create a payment on an amount $50 less than the invoiced amount.
5. In the window **Customer Open Entry Reconciliation**, fully reconcile the entries. This will give a message that the difference will be approved as a cash discount.

   This closes the payment customer entry and the invoice customer entry. Furthermore, a third customer entry is created representing the cash discount. Such an entry is always created in the closed state.

14.2.4 Blocked Entries

Customer entries may be blocked due to specific reasons. An example is that there is a dispute with the customer over an invoice. In such cases, it may not be reasonable to report on the entries. E.g., they may not be reasonable to display in AR Aging reports as these reports display the outstanding of customers. If there is a dispute over an invoice, we may not consider it an outstanding invoice we want to follow up on.

Again, consider the scenario where we have a customer entry representing an invoice, but no payment.

1. As a setup preparation, enter the window **Popup Fields** and create a new popup field named "Payment Disagreement" for the popup type "InterestReminderBlockType".
2. In the window **Change Reminder Selection**
3. Find the company customer in question. The customer entry representing the invoice appears in the lower panel.
4. Block the entry by setting the "Interest Reminder Block" field to the newly created popup field "Payment Disagreement".

   This marks the customer entry so that we can disregard it from e.g. AR Aging reports or Cash Receipt reports.

14.3 Business Layer

The universe provides measures for reporting on all sorts of registrations related to **Accounts Receivable**. There are objects both for building AR Aging reports and also for reporting on paid and invoiced amounts, or cash receipts, exchange rate differences, etc. Thereby, the universe is much broader applicable than just for **AR Aging** reports. It is also possible to report on the transactional-near debit and credit amounts and associate a large variety of dimensions.

It is only the balance objects (or objects using these in their calculation like the period objects) that really utilize the customer reconciliations and customer reconciliation journals that are related to customer entries. Other measures basically calculate entirely from fields on the customer entries. Each customer entry has a type which is stated by the field `CUSTOMERENTRY.ENTRYTYPE`. This type determines whether the entry represents an invoice, credit memo, cash discount, etc.

Certain customer entries are "born" as closed entries. An example is the entries that represent a cash discount. Customer entries have a field `OPENCLOSED` and that would actually state that entries for cash discount are indeed closed, but it not usable when working historically with entries. I.e., if we wish to provide a statement date and consider whether entries are open and closed according to that date. The entries for cash discounts are also born closed in the sense that their entry date is equal to their reconciliation

dates. Thus, no matter what statement date we select (before or after the entry's entry date), it will fall either for the requirement that it should exist on the statement date (entry date $\leq$ statement date) or it falls for the requirement that it is not yet reconciled (closing date $>$ statement date, if it were to be included in an aging report.

Queries to the AR Aging universe are likely to be combined with queries to the Finance universe to provide financial customer statements, or to the Job Cost universe to provide project related reports with a customer statement part like **Days of Sales Outstanding**.

In the following sub sections, we shall go through each of the main folders and explain the measure objects contained.

14.3.1 Debit/Credit

Objects for reporting on the individual debit and credit figures of customer entries are located in the [Debit/Credit] folder.

These objects only concern figures stored on the customer entries. No reconciliations are taken into account so it is only the pure transactional figures that are provided. There are debit and credit objects in the three currency types *Company*, *Customer*, and *Enterprise*. Furthermore, there is an object for reporting on the amount posted in the original currency of the transaction.

[Debit]	[Debit - Company], [Debit - Customer] *or* [Debit - Enterprise] *depending on the choice of currency type*
[Credit]	[Credit - Company], [Credit - Customer] *or* [Credit - Enterprise] *depending on the choice of currency type*
[Debit - Company]	[CUSTOMERENTRY.DEBITBASE]
[Credit - Company]	[CUSTOMERENTRY.CREDITBASE]
[Debit - Customer]	[CUSTOMERENTRY.DEBITCURRENCY]

[Credit - Customer]	[CUSTOMERENTRY.CREDITCURRENCY]
[Debit - Enterprise]	[CUSTOMERENTRY.DEBITENTERPRISE]
[Credit - Enterprise]	[CUSTOMERENTRY.CREDITENTERPRISE]
[Original Amount]	[CUSTOMERENTRY.ORIGINALAMOUNT]

14.3.2 Invoiced

Objects for reporting on the invoiced amounts for customers, are located in the [Invoiced] folder. To report on the invoiced amounts, we calculate the *debit – credit* for the types that are invoice related. These are the types: *Invoice*, *Credit Memo* (which is here basically just an invoice with opposed sign), and *Debit General Journal*. The last type occurs if we perform an invoice-like registration directly in the **General Journal**. The reason why entries of type *Credit General Journal* are not included is because these—by definition—are considered to represent *payments* done directly in the **General Journal**.

There are four objects for reporting on the invoiced amount; one for each of the currency types. The [Invoiced] object will prompt the user for choosing a currency type: *Customer*, *Company*, or *Enterprise*. Depending on the choice, the respective invoice object will be applied. The invoice figures include customer entries with type *Invoice*, *Credit Memo*, and *Debit General Journal*.

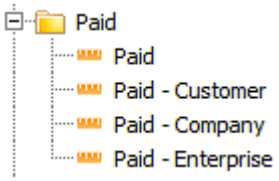
[Invoiced]	[Invoiced - Customer], [Invoiced - Company] *or* [Invoiced - Enterprise] *depending on the choice of currency type*
[Invoiced - Customer]	CUSTOMERENTRY.DEBITSTANDARD − CUSTOMERENTRY.CREDITSTANDARD *if* CUSTOMERENTRY.ENTRYTYPE ∈ (*Invoice*, *Credit Memo*, *Debit General Journal*) *(internally 0, 2, or 8)*

[Invoiced - Company]	
	CUSTOMERENTRY.DEBITBASE
–	CUSTOMERENTRY.CREDITBASE
	if CUSTOMERENTRY.ENTRYTYPE ∈ (*Invoice*, *Credit Memo*, *Debit General Journal*) *(internally 0, 2, or 8)*
[Invoiced - Enterprise]	
	CUSTOMERENTRY.DEBITENTERPRISE
–	CUSTOMERENTRY.CREDITENTERPRISE
	if CUSTOMERENTRY.ENTRYTYPE ∈ (*Invoice*, *Credit Memo*, *Debit General Journal*) *(internally 0, 2, or 8)*

14.3.3 Paid

Objects for reporting on the paid amounts for customers, are located in the [Paid] folder. To report on the paid amounts, we calculate the *credit – debit* for the entries that according to their entry type represent payments. These are the types: *Customer Payment Variance*, *Payment Variance Negative*, and *Credit General Journal*. The last type occurs if we perform an payment-like registration directly in the **General Journal**. The reason why entries of type *Debit General Journal* are not included is because these—by definition—are considered to represent *invoices* done directly in the **General Journal**.

There are four objects for reporting on the paid amount. There are one for each of the currency types. The [Paid] object will prompt the user for choosing a currency type: *Customer*, *Company*, or *Enterprise*. Depending on the choice the respective paid object will be applied.



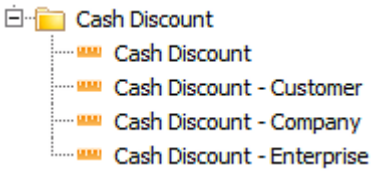
[Paid]	
	[Paid - Customer], [Paid - Company] *or* [Paid - Enterprise] *depending on the choice of currency type*
[Paid - Customer]	
	CUSTOMERENTRY.DEBITSTANDARD
–	CUSTOMERENTRY.CREDITSTANDARD
	if CUSTOMERENTRY.ENTRYTYPE ∈ (*Paid*, *Credit Memo*, *Debit General Journal*) *(internally 0, 2, or 8)*

[Paid - Company]	
	CUSTOMERENTRY.DEBITBASE
	– CUSTOMERENTRY.CREDITBASE
	if CUSTOMERENTRY.ENTRYTYPE ∈ (*Paid*, *Credit Memo*, *Debit General Journal*) *(internally 0, 2, or 8)*
[Paid - Enterprise]	
	CUSTOMERENTRY.DEBITENTERPRISE
	– CUSTOMERENTRY.CREDITENTERPRISE
	if CUSTOMERENTRY.ENTRYTYPE ∈ (*Paid*, *Credit Memo*, *Debit General Journal*) *(internally 0, 2, or 8)*

14.3.4 Cash Discounts

Objects for reporting on cash discount for customers, are located in the [Cash Discount] folder. To report on cash discount, we calculate the *credit* – *debit* for the type that represents **Cash Discount**.

There are four objects for reporting on cash discounts. There are one for each of the currency types. The [Cash Discount] object will prompt the user for choosing a currency type: *Customer*, *Company*, or *Enterprise*. Depending on the choice the respective cash discount object will be applied. The cash discount figures include customer entries with type *Cash Discount*.



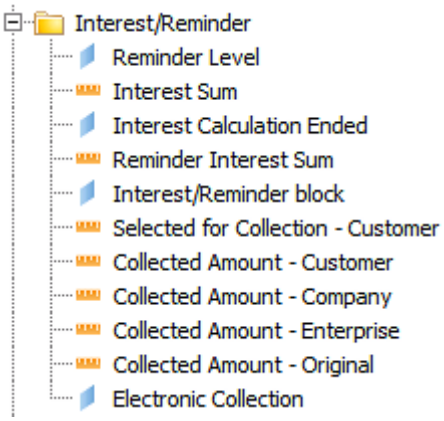
[Cash Discount]	
	[Cash Discount - Customer], [Cash Discount - Company] *or* [Cash Discount - Enterprise] *depending on the choice of currency type*
[Cash Discount - Customer]	
	CUSTOMERENTRY.DEBITSTANDARD
	– CUSTOMERENTRY.CREDITSTANDARD
	if CUSTOMERENTRY.ENTRYTYPE = *Cash Discount (internally 1)*
[Cash Discount - Company]	
	CUSTOMERENTRY.DEBITBASE
	– CUSTOMERENTRY.CREDITBASE
	if CUSTOMERENTRY.ENTRYTYPE = *Cash Discount (internally 1)*

[Cash Discount - Enterprise]	
	CUSTOMERENTRY.DEBITENTERPRISE
–	CUSTOMERENTRY.CREDITENTERPRISE
	if CUSTOMERENTRY.ENTRYTYPE = *Cash Discount (internally 1)*

14.3.5 Interests and Reminders

Objects for reporting on amounts calculated by **Maconomy** in context of interests and reminders as well as collections, are located in the [Interest and Reminders] folder. To report on these figures, we use the dedicated fields INTERESTSUM, REMINDERINTERESTSUM, COLLECTIONCURRENCY, or the BEINGCOLLECTED... fields on customer entries.

The first three objects [Interest Sum], [Reminder Interest Sum], and [Selected for Collection, Customer] provide the interest sum, the sum for interest and reminders, and the amount selected for collection in the currency of the customer. The next four objects give the amount collected in the currency stated: Customer, Company, Enterprise, or the original currency of the transaction.

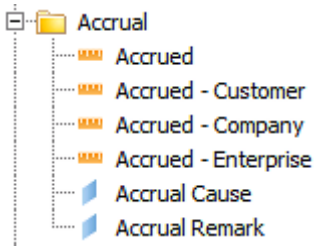


[Interest Sum]
CUSTOMERENTRY.INTERESTSUM
[Reminder Interest Sum]
CUSTOMERENTRY.REMINDERINTERESTSUM
[Selected for Collection - Customer]
CUSTOMERENTRY.COLLECTIONCURRENCY
[Collected Amount - Customer]
CUSTOMERENTRY.BEINGCOLLECTEDCURRENCY
[Collected Amount - Company]
CUSTOMERENTRY.BEINGCOLLECTEDBASE
[Collected Amount - Enterprise]
CUSTOMERENTRY.BEINGCOLLECTEDENTERPRISE
[Collected Amount - Original]
CUSTOMERENTRY.BEINGCOLLECTEDORIGINAL

14.3.6 Accrued

Objects for reporting on accrued amounts for customers, are located in the [Accrued] folder. To report on the accrued amount, we use the dedicated ACCRUEDTOTAL... fields of the customer entries.

There are four objects for reporting on accrued amounts; three for the currency types and [Accrued] which will prompt the user for selecting a currency type. Depending on the choice the respective accrue object will be applied.



[Accrued]	[Accrued - Customer], [Accrued - Company] *or* [Accrued - Enterprise] *depending on the choice of currency type*
[Accrued - Customer]	CUSTOMERENTRY.ACCRUEDTOTALCURRENCY
[Accrued - Company]	CUSTOMERENTRY.ACCRUEDTOTALBASE
[Accrued - Enterprise]	CUSTOMERENTRY.ACCRUEDTOTALENTERPRISE

14.3.7 Balances

Objects for reporting on the balance of customers, are located in the [Balances] folder. By a *balance*, we understand the summation of the balances for each customer entry (*debit–credit*), subtracting any possible reconciled amounts. The reconciled amounts are either stored on records in the table CUSTOMERRECONCILIATION or CUSTOMERRECONCILIATIONJOURNA depending on whether the customer entry in question appeared in the lower or upper panel of the window **Customer Open Entry Reconciliation**, when the reconciliation was done.

R Depending on whether the balance on the customer entry is positive or negative, we either subtract or add the reconciled amount.

Balances are always calculated according to a *statement date*. This date applies as a restriction on both customer entries and the reconciling entries in CUSTOMERRECONCILIATION and CUSTOMERRECONCILIATIONJOURNA:

- Only customer entries that *exist* on the statement date, are included. We can choose either to use the ENTRYDATE or the DUEDATE of the customer entry. Using

the ENTRYDATE, we include the customer entry if the entry date is before or equal to the statement date. If it is after, we do not consider the entry to exist yet.

- Only customer reconciliations and customer reconciliation journals that have been applied on the statement date, are included. Reconciliations actually have two dates as well:
  - the RECONCILIATIONDATE which is the date the reconciliation took place.
  - the ENTRYRECONCILIATIONDATE which is a date the user can enter to mean the date from where it should count.

  In Maconomy and BPM, we do not consider a customer reconciliation to have fully occurred until both dates have passed. This means that we only include customer reconciliations and customer reconciliation journals for which both the RECONCILIATIONDATE and ENTRYRECONCILIATIONDATE are before or equal to the statement date. If they are after the statement date, we do not consider the reconciliation to have occurred yet.

R The CUSTOMERENTRY table contains fields that state the *remainder* on the entry. However, these fields do not take any statement date into account. They state what the remainder balance is universally for the database. E.g. if we have a customer entry with an entry date of January 1^{st}, which is fully reconciled a month later, then the remainder will be zero even though we report between the creation of the entry and before the reconciliation. The old **Maconomy Analyzer** report for AR Aging use the remainder fields and would thus not always give the same results as an AR Aging report in BPM. It would not be historically correct when changing the statement date.

In Maconomy, reconciliations can also be removed. This is done if it was an error to reconcile in the first place. When a reconciliation is removed, an additional entry in the table CUSTOMERRECONCILIATIONJOURNA is created. The original entry in that table is marked with REMOVEDRECONCILIATION=1. On the corresponding entries in the table CUSTOMERRECONCILIATION the field REMOVEDRECONCILIATIONDATE are assigned a date. As we typically do not want to include reconciliations that are removed, we want only to include the customer reconciliation journals with REMOVEDRECONCILIATION=0 and the customer reconciliations with REMOVEDRECONCILIATIONDATE = NULL.

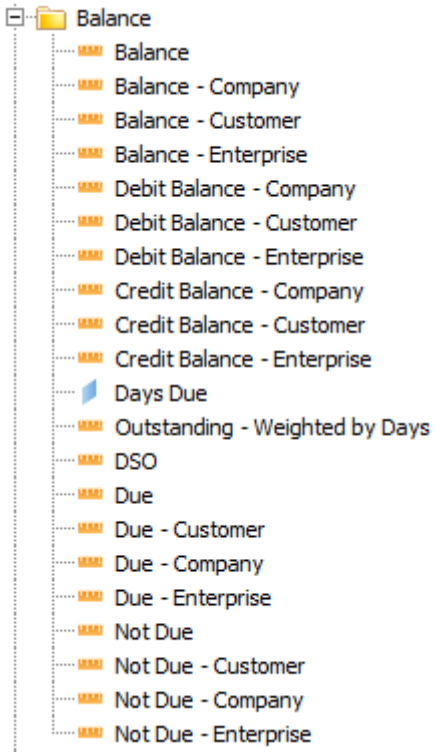
R When a reconciliation is removed in Maconomy it means that the reconciliation should never have been done. Therefore it does not make sense to report historically on when a reconciliation was removed, and BPM does not support this.

When using the **Due** og **Not Due** objects in the [Balance] folder, or when using the objects in the [Aging] folder, an **Aging Principle** needs to be applied. This aging principle defines whether the statement date is using the entry date or the due date of customer entries. It also defines the periods into which amounts should be put when reporting on the age of the entries. When just using the normal balance objects in the [Balance]

folder, usually a restriction on statement date should apply; i.e. if we want the balance to be as of that statement date. Else the balance will be what is ultimately stored in the database without any date consideration. For more information about the aging principles, see Section 14.3.8.

The [`Balances`] folder contains three kinds of objects:

- The balance objects. There is one for each of the three currency types: *Customer*, *Company* and *Enterprise*. There is also an object [`Balance`] which—depending on the choice of currency type—issues one of the three currency type specific balance objects. Furtermore, there are debit/credit versions of the balance that only states the debit and credit amount, respectively.
- The [`Outstanding - Weighted by Days`] and the [`DSO`] objects. The first calculates a summation of the balances of the customer entries and for each customer entry, it *weighs* (multiplies) with the number of days the entry has been due. If we apply the *due date* in the aging principle, we calculate from the statement date to the due date. The object can be used as an indication of which customers have high amounts of outstandings or outstandings that are old. Both cases are problematic from a cashflow point of view. However, the preferred object to use is the [`DSO`]. This object divides the result of the [`Outstanding - Weighted by Days`] with the total balance of the entries. This object is an indicator of unit *Time*. A high value for a customer indicates that for this customer we have high or old outstandings compared to other customers. The value in it self does not mean anything, but if we compare the value for different customers, it clearly outlines which customers to focus on in a follow up.
- The due/not due objects are special versions of the balance objects, where the aging principle and statement date are used for determining the balance that is due (i.e. past statement date) and not due, respectively.



[Balance] [Balance - Company], [Balance - Customer] *or* [Balance - Enterprise] *depending on the choice of currency type*
[Balance - Company] CUSTOMERENTRY.DEBITBASE − CUSTOMERENTRY.CREDITBASE − $\sum$ CUSTOMERRECONCILIATION.RECONCILEDLINEBASE − $\sum$ CUSTOMERRECONCILIATIONJOURNA.RECONCILINEDLINEBASE *if* CUSTOMERENTRY.DEBITBASE −CUSTOMERENTRY.CREDITBASE > 0 CUSTOMERENTRY.DEBITBASE − CUSTOMERENTRY.CREDITBASE + $\sum$ CUSTOMERRECONCILIATION.RECONCILEDLINEBASE + $\sum$ CUSTOMERRECONCILIATIONJOURNA.RECONCILINEDLINEBASE *if* CUSTOMERENTRY.DEBITBASE −CUSTOMERENTRY.CREDITBASE ≤ 0
[Balance - Customer] CUSTOMERENTRY.DEBITSTANDARD − CUSTOMERENTRY.CREDITSTANDARD − $\sum$ CUSTOMERRECONCILIATION.RECONCILEDLINESTANDARD − $\sum$ CUSTOMERRECONCILIATIONJOURNA.RECONCILINEDLINESTANDARD *if* CUSTOMERENTRY.DEBITSTANDARD −CUSTOMERENTRY.CREDITSTANDARD > 0 CUSTOMERENTRY.DEBITSTANDARD − CUSTOMERENTRY.CREDITSTANDARD + $\sum$ CUSTOMERRECONCILIATION.RECONCILEDLINESTANDARD + $\sum$ CUSTOMERRECONCILIATIONJOURNA.RECONCILINEDLINESTANDARD *if* CUSTOMERENTRY.DEBITSTANDARD −CUSTOMERENTRY.CREDITSTANDARD ≤ 0

[Balance - Enterprise]

$$
\begin{aligned}
& \texttt{CUSTOMERENTRY.DEBITENTERPRISE} \\
- \; & \texttt{CUSTOMERENTRY.CREDITENTERPRISE} \\
- \; & \sum \texttt{CUSTOMERRECONCILIATION.RECONCILEDLINEENTERPRISE} \\
- \; & \sum \texttt{CUSTOMERRECONCILIATIONJOURNA.RECONCILINEDLINEENTERPRISE} \\
& \textit{if } \texttt{CUSTOMERENTRY.DEBITENTERPRISE} \\
& -\texttt{CUSTOMERENTRY.CREDITENTERPRISE} > 0
\end{aligned}
$$

$$
\begin{aligned}
& \texttt{CUSTOMERENTRY.DEBITENTERPRISE} \\
- \; & \texttt{CUSTOMERENTRY.CREDITENTERPRISE} \\
+ \; & \sum \texttt{CUSTOMERRECONCILIATION.RECONCILEDLINEENTERPRISE} \\
+ \; & \sum \texttt{CUSTOMERRECONCILIATIONJOURNA.RECONCILINEDLINEENTERPRISE} \\
& \textit{if } \texttt{CUSTOMERENTRY.DEBITENTERPRISE} \\
& -\texttt{CUSTOMERENTRY.CREDITENTERPRISE} \leq 0
\end{aligned}
$$

[Debit Balance - Company]

$$
\begin{aligned}
& \texttt{CUSTOMERENTRY.DEBITBASE} \\
- \; & \sum \texttt{CUSTOMERRECONCILIATION.RECONCILEDLINEBASE} \\
- \; & \sum \texttt{CUSTOMERRECONCILIATIONJOURNA.RECONCILINEDLINEBASE} \\
& \textit{if } \texttt{CUSTOMERENTRY.DEBITBASE} > 0 \\
& 0 \textit{ otherwise}
\end{aligned}
$$

[Debit Balance - Customer]

$$
\begin{aligned}
& \texttt{CUSTOMERENTRY.DEBITSTANDARD} \\
- \; & \sum \texttt{CUSTOMERRECONCILIATION.RECONCILEDLINESTANDARD} \\
- \; & \sum \texttt{CUSTOMERRECONCILIATIONJOURNA.RECONCILINEDLINESTANDARD} \\
& \textit{if } \texttt{CUSTOMERENTRY.DEBITSTANDARD} > 0 \\
& 0 \textit{ otherwise}
\end{aligned}
$$

[Debit Balance - Enterprise]

$$
\begin{aligned}
& \texttt{CUSTOMERENTRY.DEBITENTERPRISE} \\
- \; & \sum \texttt{CUSTOMERRECONCILIATION.RECONCILEDLINEENTERPRISE} \\
- \; & \sum \texttt{CUSTOMERRECONCILIATIONJOURNA.RECONCILINEDLINEENTERPRISE} \\
& \textit{if } \texttt{CUSTOMERENTRY.DEBITENTERPRISE} > 0 \\
& 0 \textit{ otherwise}
\end{aligned}
$$

[Credit Balance - Company]

$$
\begin{aligned}
& \texttt{CUSTOMERENTRY.CREDITBASE} \\
- \; & \sum \texttt{CUSTOMERRECONCILIATION.RECONCILEDLINEBASE} \\
- \; & \sum \texttt{CUSTOMERRECONCILIATIONJOURNA.RECONCILINEDLINEBASE} \\
& \textit{if } \texttt{CUSTOMERENTRY.CREDITBASE} > 0 \\
& 0 \textit{ otherwise}
\end{aligned}
$$

[Credit Balance - Customer]	CUSTOMERENTRY.CREDITSTANDARD $-$ $\sum$ CUSTOMERRECONCILIATION.RECONCILEDLINESTANDARD $-$ $\sum$ CUSTOMERRECONCILIATIONJOURNA.RECONCILINEDLINESTANDARD *if* CUSTOMERENTRY.CREDITSTANDARD > 0 0 *otherwise*
[Credit Balance - Enterprise]	CUSTOMERENTRY.CREDITENTERPRISE $-$ $\sum$ CUSTOMERRECONCILIATION.RECONCILEDLINEENTERPRISE $-$ $\sum$ CUSTOMERRECONCILIATIONJOURNA.RECONCILINEDLINEENTERPRISE *if* CUSTOMERENTRY.CREDITENTERPRISE > 0 0 *otherwise*
[Outstanding - Weighted by Days]	(CUSTOMERENTRY.DEBITBASE $-$ CUSTOMERENTRY.CREDITBASE) $\times$ $days_\Delta$ *where* $days_\Delta$ = *daysbetween*([Statement Date], [date]) *and* [date] *is entry date or due date (chosen by the user)*
[DSO]	$\frac{\text{[Outstanding - Weighted by Days]}}{\sum (\text{CUSTOMERENTRY.DEBITBASE} - \text{CUSTOMERENTRY.CREDITBASE})}$
[Due - Customer]	[Balance - Customer] *if* [Aging Units Due] > 0
[Due - Company]	[Balance - Company] *if* [Aging Units Due] > 0
[Due - Enterprise]	[Balance - Enterprise] *if* [Aging Units Due] > 0
[Not Due - Customer]	[Balance - Customer] *if* [Aging Units Due] ≤ 0
[Not Due - Company]	[Balance - Company] *if* [Aging Units Due] ≤ 0
[Not Due - Enterprise]	[Balance - Enterprise] *if* [Aging Units Due] ≤ 0

In the above table, records in the CUSTOMERENTRY table are actually paired with records in the CUSTOMERRECONCILIATION and CUSTOMERRECONCILIATIONJOURNA tables as they both may contribute with amounts that are reconciled against the balances from the records in the CUSTOMERENTRY table.

Statement Date is the date provided by the user when running the aging reports. The function *daysbetween* gives the number of days between the arguments, if positive; zero otherwise. The object [Aging Date] is either the *ce*.ENTRYDATE or *ce*.DUEDATE depending on the chosen **Aging Principle**.

14.3.8 Aging

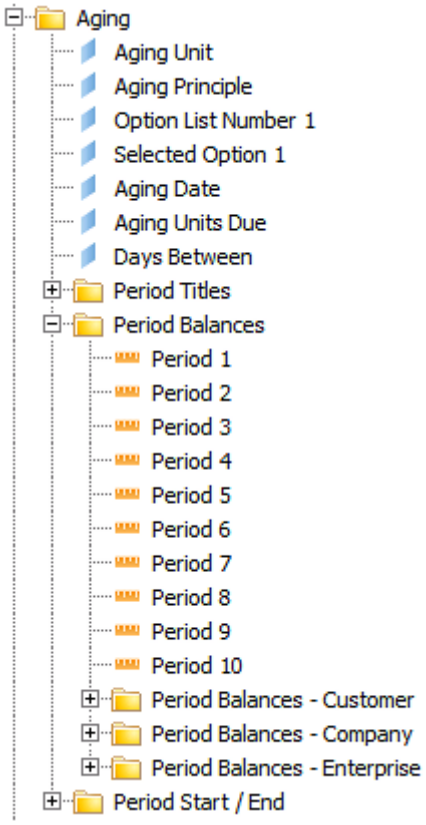
The objects for reporting on aged customer amounts are located in the folder [`Aging`]. An aged amount is a customer entry balance that is categorized to belong to one of several aging periods. The aging periods are defined by an **Aging Principle** in Maconomy. Each aging principle states whether to use the entry date or the due date, and it states whether periods are defined by days or months. Furthermore, an aging principle contains a collection of up to 10 periods. Each period states a *from* and a *to* limit but the period can also be open. E.g. we can define a period from 30 days and beyond. Each period is assigned a text; e.g. `30+` or `0-29`.

In BPM Reporting, we need to determine the limits of the aging periods in order to find out which period each customer entry (or reconciliation) belongs to. To do so, we first calculate the lower and upper limits of each period. These are represented as integers. If the aging principle is day-based the integers count the number of days from the statement date. If it is month-based the integers count the number of months from the statement date. If the interval is open, we use the value 9999 or -9999; depending on whether it is an upper limit or a lower limit; and depending on whether the period is looking ahead or backwards. When we have all the limits, we can determine whether a balance belongs to one of these.

There are 10 period objects that will prompt the user for choosing a currency type: *Customer*, *Company*, or *Enterprise*. Furthermore, there are 10 measure objects for each of the three currency types. Below, we have stated the calculations of these measure objects assuming that they are in the currency of the company.

The folder [`Period Titles`] contains 10 dimension objects stating the chosen title for each of the 10 periods. These are convenient to use as headers in report tables.

The folder [`Period Start/End`] contains the dimension objects that calculate the lower and upper period limits. These can be convenient if defining new kinds of period objects. Furthermore, we have a couple of dimensions for determining the aging principle (the objects [`Aging Principle`], [`Option List Number 1`], and [`Selected Option 1`].) The object [`Aging Date`] states either the entry date or the due date, depending on what is set for the aging principle. The object [`Aging Units Due`] is a helper-object that states the number of units (days or months) that an entry is due. A positive number means due (i.e., passed the statement date), zero means equal to the statement date and negative means not due. The object [`Days Between`] is another helper-object that states the number of days between the entry and the statement date.



[`Period 1`]	[`Balance - Company`] *if* [`Aging Units Due`] $\in$ [[`Period 1 Start`]; [`Period 1 End`]]
[`Period 2`]	[`Balance - Company`] *if* [`Aging Units Due`] $\in$ [[`Period 2 Start`]; [`Period 2 End`]]
[`Period 3`]	[`Balance - Company`] *if* [`Aging Units Due`] $\in$ [[`Period 3 Start`]; [`Period 3 End`]]

[Period 4]	[Balance - Company] *if* [Aging Units Due] ∈ [[Period 4 Start]; [Period 4 End]]
[Period 5]	[Balance - Company] *if* [Aging Units Due] ∈ [[Period 5 Start]; [Period 5 End]]
[Period 6]	[Balance - Company] *if* [Aging Units Due] ∈ [[Period 6 Start]; [Period 6 End]]
[Period 7]	[Balance - Company] *if* [Aging Units Due] ∈ [[Period 7 Start]; [Period 7 End]]
[Period 8]	[Balance - Company] *if* [Aging Units Due] ∈ [[Period 8 Start]; [Period 8 End]]
[Period 9]	[Balance - Company] *if* [Aging Units Due] ∈ [[Period 9 Start]; [Period 9 End]]
[Period 10]	[Balance - Company] *if* [Aging Units Due] ∈ [[Period 10 Start]; [Period 10 End]]
[Period 11]	[Balance - Company] *if* [Aging Units Due] ∈ [[Period 11 Start]; [Period 11 End]]
[Period 12]	[Balance - Company] *if* [Aging Units Due] ∈ [[Period 12 Start]; [Period 12 End]]

14.4 Reporting Examples

14.4.1 Customer Balances

By a **Customer Balance**, we understand the amount that in total has been invoiced customers but not yet paid or reconciled. So this is really the outstanding balance on customers. In context of customer balances, we do not consider customer payments to count until they are reconciled. If a payment is engaged in a partial reconciliation, only the reconciled amount is included in the calculation of the outstanding amount.

Example 14.1 (Reporting on customer balances) *In this first example, we shall create a report that outlines the outstanding balance of customers.*

1. *Create a new WebIntelligence document and select the universe* AR Aging

2. *Add the following objects for the query:*

   [Currency] *(folder [Currency])*

   [Customer No.] *(folder [Company Customer])*

   [Customer Name] *(folder [Company Customer])*

   [Balance] *(folder [Balance])*

3. *Run the report*

The report will prompt the user for selecting in which currency, the outstanding amount should be shown. The report shows a table of four columns: Currency, customer number, customer name, and the outstanding amount of the customer displayed on that row. For convenience, you may drag the currency column up above the table to create a section on currencies.

In this report, we will get all the customers and their outstandings. It uses whatever data is in the Maconomy database, but is not working historically with regards to invoiced amounts nor paid amounts. So if we have an invoice that was created before the date the report is run but paid and reconciled with entry reconciliation date in the future (which is possible), the report will still pick up that reconciliation. Usually, we want calculated balances as of a statement date provided by the use. This we shall see in the next example. ■

14.4.2 Outstandings as of a statement date

For reports that display the outstanding of customers, the normal approach is to calculate these as of a given statement date. Typically, that statement date is one the user selects when running the report or it could be the current date.

Example 14.2 (Customer outstandings as of a statement date) *Consider the report we created in Example 14.1 and do the following enhancements to it:*

1. *To the following changes to the report:*
   - *Add the object [Statement Date] (folder [Date]) to the query.*
   - *Add a filter on the object [Closing Date] and require that it is Greater Than a prompt value with title "Statement Date", or Null (i.e., we need two filters with a disjunction between.)*
   - *Replace the object [Balance] with the object [Due]*
2. *Run the report*

In addition to prompt for a currency type, the report will prompt for a statement date. The object [Statement Date] takes care of that. The [Due] object applies a filter such that we only get the balance as of that statement date.

The filter on [Closing Date] is barely a performance ptimization. It excludes all the records that are closed at statement date. This means customer entries that on the statement date, are fully reconciled. By fully reconciled, we understand that both the reconciliation and entry reconciliation dates have passed. When we then require that the closing date is after the statement date or null, it means that we include only the records where the reconciliation has not been doned fully yet (according to the statement date) or not done at all. The rest of the objects are those that on the statement date or before have been fully reconciled and these would not contribute to the balance anyway. We get a performance improvement by excluding these entries because they usually make up the far majority of the entries. ■

14.4.3 Aging Balances

The age of an outstanding amount may have influence on the likelyhood of whether to actually receive the payment for the job. Therefore, we often wish to categorize such balances into aging periods. The older an outstanding amount is, the more we should doubt receiving the payment.

Example 14.3 (Aging outstanding balances) *Enhance the report we did in Example 14.2, as follows:*

1. *Remove the [Due] object. It is not needed anymore.*
2. *Add the following objects to the query:*

   [Period 1] *(folder [Period Balances])*

   [Period 2] *(folder [Period Balances])*

   [Period 3] *(folder [Period Balances])*

   [Period 4] *(folder [Period Balances])*

   [Period 1 Title] *(folder [Period Titles])*

   [Period 2 Title] *(folder [Period Titles])*

   [Period 3 Title] *(folder [Period Titles])*

   [Period 4 Title] *(folder [Period Titles])*
3. *Add the following filters to the query:*
   - *A filter on the object [[]From Date] (folder [Dates]). Require that it is Less Than Or Equal to [Statement Date]*
   - *A filter on the object [Option List Number 1] (folder [Aging]). Set it to equal "Aging Principles"*

- *A filter on the object [`Selected Option 1`] (folder [`Aging`]). Set it to equal `"AR Aging"`*

4. *Add the period objects as new columns and use the period title objects as column headers.*

The new filter on [`From Date`] is crucial because it makes sure that we only include entries and reconciliations that are existing at the statement date. The report assumes that an aging principle that references an option list named `"Aging Principles"` and selects an option named `AR Aging`, has been created in Maconomy. They are usually part of a standardized setup.

The outstanding balances will be categorized into the four periods that we defined. However, notice that some of the aging periods could state not due *balances. It depends entirely on the aging principles. Thus, it is very easy to create AR Aging reports that are completely flexible to data. The report will prompt for whether we want to filter entries by entry date or due date. The aging principle determines whether we categorise the included entries after the entry date or due date. Periods in the aging principle can be either due or not due. Typically, entries fall in the Due category, but there are cases where the not due category applies. If the report is filtering on entry date and the aging principle age by due date, then all the included entries that have a due date after the statement date, are considered* not due. *If we filter by due date or the aging principle categorises by entry date, we can only get entries in the* due *category. This report is actually a simple (but fully functional)* ***AR Aging report***. ■

14.4.4 Filtering blocked and inactive customers

It is easy to filter off blocked or inactive customers from AR Aging reports. Typically, these customers are handled in another workflow than the one where AR Aging reports are applied. E.g. blocked customers may be catched up using various kinds of bad debt reporting and may just blur the picture if included in an AR Aging report. Similar goes for inactive customers.

Example 14.4 (Filter off blocked and inactive customers) *To filter off blocked and inactive customers, we simply add the following filters (see one of the previous reports in the chapter as basis):*

[`Include Blocked`] *(folder [`Entry Information`])*

[`Include Inactive Customers`] *(folder [`Allow Use`])*

The report will now prompt for whether to include blocked entries and whether to include inactive customers, respectively. ■

14.5 Data Foundation

14.5.1 Fact Tables

The AR Aging universe is centered around a single fact table CUSTOMERENTRYREMAINDER_D.

CUSTOMERENTRYREMAINDER_D

The fact table CUSTOMERENTRYREMAINDER_D combines rows in CUSTOMERENTRY representing invoices, payments, cash discounts, etc., with rows in the tables CUSTOMERRECONCILIATION and CUSTOMERRECONCILIATIONJOURNA which store values with which we have reconciled the customer entries.

The structure of the fact table is two unions between selects to the following:

- CUSTOMERENTRY. This select provides the measures and dimensions for invoices, payments, cash discounts, etc.
- a join between CUSTOMERENTRY and CUSTOMERRECONCILIATION. This select provides the figures we have reconciled from the table part in the customer reconciliation window of Maconomy.
- a join between CUSTOMERENTRY and CUSTOMERRECONCILIATIONJOURNA. This select provides the figures we have reconciled from the card part in the customer reconciliation window of Maconomy.

Using this fact, we are able to define the balance of customer entries. Such balances can then be restricted by their entry date or due date, and they can be aged by the same sorts of dates. It also provides the foundation for other kinds of measures like paid, cash discount, and similar figures that come out of the fields of customer entries. However, for reporting on paid and invoiced amounts, it may be beneficial to use the universe Customer Payment instead.

Chapter 15

Customer Payment Universe

The Customer Payment universe provides objects for reporting on customer payments and the invoices against which these payments have been reconciled. Concerning payments, it is possible to report on **Paid Amounts**, **Cash Discount**, and **Payment Variance**. It is also possible to report on the amounts that have been reconciled against payments. Concerning invoices, it is possible to report on the **Invoiced Amount** including or excluding taxes, charges, **Invoice Discount** given, and the reduction on possible amounts invoiced on account. It is also possible to report on the amounts that have been reconciled against invoices.

The paid and invoiced figures can be reported on individually while still being able to tie specific payment amounts to individual invoices; also when e.g. multiple payments cross-wise have been reconciled against multiple invoices. E.g. it is possible to report the individual paid amounts from one customer payment that has been reconciled (possibly partly) against two invoices; including individual cash discounts, etc. This ability makes the universe suitable for any kind of cash receipts, customer payment, or invoice status reporting.

The paid and invoiced amounts can be grouped and restricted by a variety of dimensions including payment and invoice dates, standard dimensions, company and customer, as well as job and customer hierarchies. It is also possible to report on various transactional dimensions used for identifying and slicing the payments and invoices.

15.1 Prerequisites

In order to fully benefit from the Customer Payment universe, Maconomy should be set up such that:

- The system parameter **Use Calendar Day** is enabled and the calendar spans all necessary dates.

15.2 Maconomy Workflows

In the following sub-sections, we shall look at some common Maconomy workflows that produce the data which the Customer Payment universe is made for reporting on. First, we shall look at a simple workflow where an invoice is reconciled against a customer payment. Then we shall look at a more complex variant where multiple payments partially cover the amounts of multiple invoices; including some cash discounts. Finally, we shall look at a workflow including an invoice on account followed by a time and material invoice thereby having an account reduction, and the payment associated.

15.2.1 Customer Payment of Invoices

In Maconomy, customer invoices are represented in various ways. The main table representing a whole invoice is the `INVOICE` table. All sorts of invoices—including on account invoices—are represented by this table. However, the various transactions related to *accounts receivable* also result in entries in the `CUSTOMERENTRY` table. Both invoices, payments, cash discounts, etc. are represented by entries in the `CUSTOMERENTRY` table. In the AR Aging universe all figures are taken from records in that table and the two reconciliation tables `CUSTOMERRECONCILIATION` and `CUSTOMERRECONCILIATIONJOURNAL`. In the Customer Payment universe, invoice-related data are found in records of the `INVOICE` table, while payment-related data are found in records of the tables `CUSTOMERENTRY` and `CUSTOMERRECONCILIATION`. The latter table binds customer entries pair-wise; one entry often representing an invoice (or credit memo) and the other often representing a payment or other entry reconciled against the invoice; although, more complex scenarios also exists.

Consider the following flow:

1. In the window **Jobs**, we create a new job.
2. In the window **Job Journal**, we create a new journal. On a line we enter the number of the new job, an invoiceable time activity and a quantity.
3. We post the journal.
4. In the window **Invoice Selection**, we approve the invoice selection and print the invoice.

   This creates an `INVOICE` record along with an entry in the table `CUSTOMERENTRY`.
5. In the window **Customer Payments**, we create a new journal stating a payment amount corresponding to the invoiced amount.
6. In the window **Customer Open Entry Reconciliation**, we browse for the payment in the upper panel and see the corresponding customer entry record representing the invoice.
7. We reconcile the two entries fully.

This provides a `CUSTOMERRECONCILIATION` *record and a* `CUSTOMERRECONCILIATION-JOURNAL` *record* in addition to a record in the tables `CUSTOMERENTRY` and `CUSTOMERPAYMENT`.

The `INVOICE` record states the different information we wish to report on related to the invoice:

- the invoiced amount
- the tax amount
- whether it is an invoice on account or a time and material invoice.
- any charges, discounts, etc.

The two `CUSTOMERENTRY` records are tied together by the `CUSTOMERRECONCILIATION` record. This structure makes it possible to report on what has individually been reconciled (paid, given in cash discount, etc.) for each payment to the present invoice. E.g. the structure states:

- the amount paid related to that invoice.
- the amount given in cash discount.
- payment variance like exchange rate gain or losses.

The paid amount is actually the amount paid related to a specific invoice. Thereby, it is possible to distinguish the different amounts that a single payment reconciles against multiple invoices. Thereby, we are both able to handle single payments that each reconcile against multiple invoices, and multiple payments that reconcile against single invoices. This we shall dive further into below.

15.2.2 Multiple Payments against Multiple Invoices

To fully see the strength of the universes facts, consider the following flow where we create two payments against three invoices.

1. In the window **Jobs**, create three jobs numbered 2030, 2040, and 2050.
2. In the window **Job Journal**, create three journals stating invoiceable activities and non-zero quantities.
3. In the window **Invoice Selection**, approve and print the invoice for each job.

   This provides three new records in the `INVOICE` *table.*
4. In the window **Customer Payments** create two new customer payments. The first payment should cover the first invoice and part of the second invoice. The second payment should cover the rest of the second invoice and 5% less the full amount of the third invoice.

5. In the window **Customer Open Entry Reconciliation** locate the first customer payment.
6. Reconcile the payment fully against the first two invoices.

   This provides two new records in the table `CUSTOMERRECONCILIATION`*. One states what the payment reconciles against the first invoice. The other states what the payment reconciles against the second invoice. A record was also created in the table* `CUSTOMERRECONCILIATIONJOURNAL`*. This record states what has been reconciled against the payment. Actually, this system is more general. The record that the customer reconciliation journal relates to could also have been a credit memo, and the records that the customer reconciliation relates to could also have been payments. This is because the customer reconciliation journal is stating what has been reconciled against the record chosen in the upper panel of the* ***Customer Open Entry Reconciliation*** *window of Maconomy, whereas customer reconciliation records have to do with the lower panel. Therefore, when we look at what has been reconciled against customer entries in general, we need to look at records in both reconcilation tables.*
7. Now, browse to the other payment.
8. Reconcile that payment fully against the second and third invoice.

   This provides yet another two records in the table `CUSTOMERRECONCILIATION` *and a record in* `CUSTOMERRECONCILIATIONJOURNAL`*.*

From the entries, we notice the following:

- A cash discount has been allocated for the second reconciliation.
- The second invoice is paid with contribution from both the first and second payment.
- We can report on the full invoiced amounts, the full paid amounts, but also the amounts that each customer payment individually reconciles against each invoice.

15.2.3 On Account Reduction

Consider the flow where we first invoice on account and then follow with a normal time & material invoice.

1. In the window **Jobs** create a new job numbered 3010
2. In the window **Invoice Selection** enter an amount of $1000 for invoicing on account for the job.
3. Approve and print the invoice.
4. In the window **Job Journal** create a new journal.

5. Enter a line stating the job, an invoiceable activity and a quantity resulting in a billing amount of $1300.[1]

6. In the window **Invoice Selection** approve and print the invoice of the work.

   We now have two records in the `INVOICE` *table. One for the invoice on account and one for the time and material invoice. The former has an invoice amount of* $1000*. The latter has an invoice amount of* $200 *and states an* ***On Account Reduction*** *of* $1000*.*

7. In the window **Customer Payments** create a payment of $1300.

8. In the window **Customer Open Entry Reconciliation** reconcile the customer payment against the invoices.

   This will provide two records in `CUSTOMERRECONCILIATION`*. The former record associates the payment to the invoice on account, stating that an amount of* $1000 *has been reconciled. The latter record associates the payment to the time and material invoice, stating that an amount of* $200 *has been reconciled.* There is of course also a record in `CUSTOMERRECONCILIATIONJOURNAL`.

15.3 Business Layer

15.3.1 Paid amounts

The measure objects for reporting on customer payments, are located in the [`Paid`] folder. There are four sorts of payment measures:

Paid amounts which are the actual amounts paid by customers.

Reconciled amounts which are the amounts reconciled

Cash Discounts which are the amounts given in cash discount.

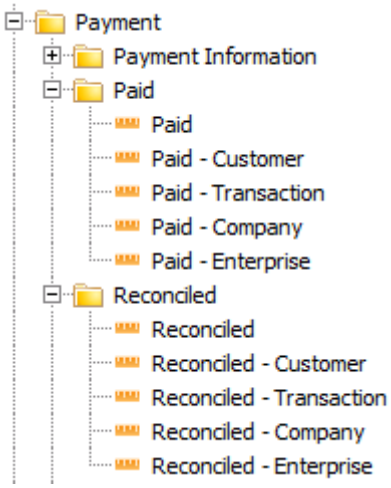
Payment Variances which are the amounts stating variances like exchange gains or losses.

The universe favors flexibility in reporting. It does by offering the ability to report on reconciled amounts against any sort of customer entry; not just against invoices.

The reconciled amounts are the amounts that have been reconciled against any customer entry. Thus, they are not just amounts reconciled against invoices or against payments. Therefore—when reporting on reconciled amounts—it is highly important to filter by some dimensions like [`Entry Type`]. The reason is because the reconciled amounts are taken from the two reconciliation records. The same goes for cash discount and payment variance measures.

[1] We assume that the amounts mentioned are the only amounts involved and that we do not need to assume invoice discounts, taxes, etc.

The four kinds of measure objects are available in four currency types: *Customer*, *Transaction*, *Company*, and *Enterprise*. The paid amounts are rooted in customer entries, whereas reconciled amounts, cash discounts and payment variance are figures fetched on the individual line level; i.e. from the reconciliations. To the right, we only outline the paid and reconcile-related measures.

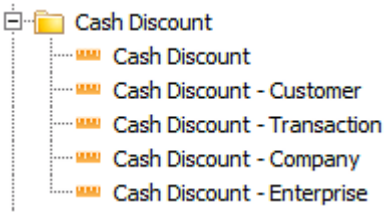


R Note, that the company currency of an invoice to a foreign customer typically differs from the company currency of the corresponding payment. The invoice is typically made in the base company whereas the payment is in the company customer's own currency. Thus, reconciled amounts should not be compared or balances made to payment related amounts like paid, cash discount, etc. This is not the case when reporting in enterprise, customer or transaction currency. Here, currencies always correspond.

[Paid]	[Paid - Customer], [Paid - Transaction], [Paid - Company], or [Paid - Enterprise] *depending on the choice of currency type*
[Paid - Customer]	CUSTOMERENTRY.DEBITSTANDARD − CUSTOMERENTRY.CREDITSTANDARD *if* CUSTOMERENTRY.ENTRYTYPEPN = 9 CUSTOMERENTRY.CREDITSTANDARD − CUSTOMERENTRY.DEBITSTANDARD *if* CUSTOMERENTRY.ENTRYTYPEPN ∈ (7, 10)
[Paid - Transaction]	CUSTOMERENTRY.ORIGINALAMOUNT *if* CUSTOMERENTRY.ENTRYTYPEPN = 9 −CUSTOMERENTRY.ORIGINALAMOUNT *if* CUSTOMERENTRY.ENTRYTYPEPN ∈ (7, 10)
[Paid - Company]	CUSTOMERENTRY.DEBITBASE − CUSTOMERENTRY.CREDITBASE *if* CUSTOMERENTRY.ENTRYTYPEPN = 9 CUSTOMERENTRY.CREDITBASE − CUSTOMERENTRY.DEBITBASE *if* CUSTOMERENTRY.ENTRYTYPEPN ∈ (7, 10)

[Paid - Enterprise] CUSTOMERENTRY.DEBITENTERPRISE −CUSTOMERENTRY.CREDITENTERPRISE *if* CUSTOMERENTRY.ENTRYTYPEPN = 9 CUSTOMERENTRY.CREDITENTERPRISE −CUSTOMERENTRY.DEBITENTERPRISE *if* CUSTOMERENTRY.ENTRYTYPEPN ∈ (7, 10)
[Reconciled - Customer] CUSTOMERRECONCILIATION.RECONCILEDLINESTANDARD +CUSTOMERRECONCILIATIONJOURNA.RECONCILEDLINESTANDARD
[Reconciled - Transaction] CUSTOMERRECONCILIATION.RECONCILEDLINEORIGINAL +CUSTOMERRECONCILIATIONJOURNA.RECONCILEDLINEORIGINAL
[Reconciled - Company] CUSTOMERRECONCILIATION.RECONCILEDLINEBASE +CUSTOMERRECONCILIATIONJOURNA.RECONCILEDLINEBASE
[Reconciled - Enterprise] CUSTOMERRECONCILIATION.RECONCILEDLINEENTERPRISE +CUSTOMERRECONCILIATIONJOURNA.RECONCILEDLINEENTERPRISE

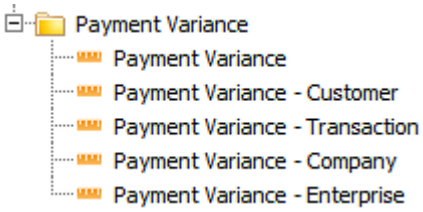
The measures for cash discounts are available in the same four currencies as the *Paid* amount objects. The same considerations concerning comparing to reconciled amounts are valid for cash discount objects. The cash discount objects are rooted in the customer reconciliation records.



[Cash Discount - Customer] CUSTOMERRECONCILIATION.CASHDISCOUNTSTANDARD +CUSTOMERRECONCILIATIONJOURNA.CASHDISCOUNTSTANDARD
[Cash Discount - Transaction] CUSTOMERRECONCILIATION.CASHDISCOUNTORIGINAL +CUSTOMERRECONCILIATIONJOURNA.CASHDISCOUNTORIGINAL
[Cash Discount - Company] CUSTOMERRECONCILIATION.CASHDISCOUNTBASE +CUSTOMERRECONCILIATIONJOURNA.CASHDISCOUNTBASE

[Cash Discount - Enterprise]	
	CUSTOMERRECONCILIATION.CASHDISCOUNTENTERPRISE
	+CUSTOMERRECONCILIATIONJOURNA.CASHDISCOUNTENTERPRISE

The measures for payment variances are available in the same four currencies as the *Paid* amount objects. The same considerations concerning comparing to reconciled amounts are valid for payment variance objects. The payment variance objects are rooted in the customer reconciliation records.



[Payment Variance - Customer]	
	CUSTOMERRECONCILIATION.CUSTPAYMENTVARIANCESTANDARD
	+CUSTOMERRECONCILIATIONJOURNA.CUSTPAYMENTVARIANCESTANDARD
[Payment Variance - Transaction]	
	CUSTOMERRECONCILIATION.CUSTPAYMENTVARIANCEORIGINAL
	+CUSTOMERRECONCILIATIONJOURNA.CUSTPAYMENTVARIANCEORIGINAL
[Payment Variance - Company]	
	CUSTOMERRECONCILIATION.CUSTPAYMENTVARIANCEBASE
	+CUSTOMERRECONCILIATIONJOURNA.CUSTPAYMENTVARIANCEBASE
[Payment Variance - Enterprise]	
	CUSTOMERRECONCILIATION.CUSTPAYMENTVARIANCEENTERPRIS
	+CUSTOMERRECONCILIATIONJOURNA.CUSTPAYMENTVARIANCEENTERPRIS

15.3.2 Invoiced amounts

The measure objects for reporting on invoiced amounts, are located in the [Invoiced] folder. There are three sorts of measures:

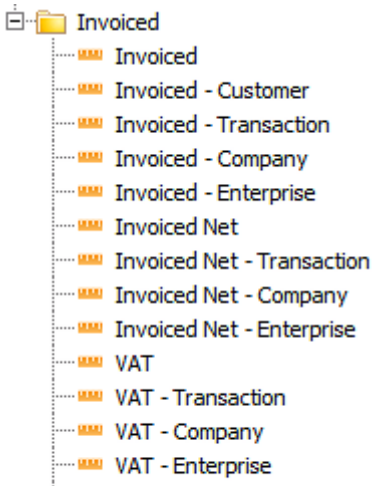
Invoiced which is the invoiced amount incl. Tax.

Invoiced Net which is the invoiced amount excl. Tax.

Tax which is the Tax amount of the invoiced amount.

R For all invoice-related measures, we test on the field EXINVOICE.DEBITCREDITPN. If the value is not 1, it is because we have a credit memo and in such cases, we negate the value represented by the database field.

All the measures are rooted in fields from the `CUSTOMERENTRY` for entries that represent the reconciling invoice of payments.

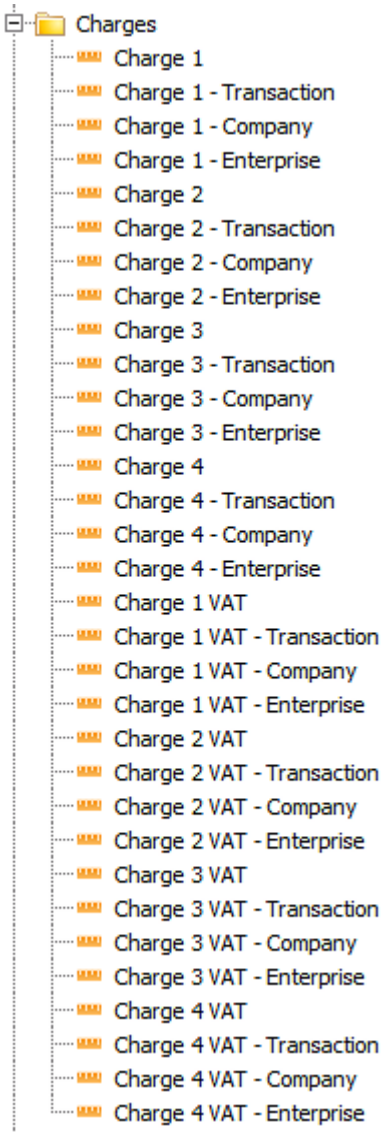


[Invoiced]	`INVOICE.TOTALCURRENCY`, `INVOICE.TOTALBASE`, *or* `INVOICE.TOTALENTERPRISE` *depending on the user's choice of Currency Type*
[Invoiced - Customer]	`INVOICE.TOTALSTANDARD`
[Invoiced - Transaction]	`INVOICE.TOTALCURRENCY`
[Invoiced - Company]	`INVOICE.TOTALBASE`
[Invoiced - Enterprise]	`INVOICE.TOTALENTERPRISE`
[Invoiced Net]	`INVOICE.ITEMSUMCURRENCY`, `INVOICE.ITEMSUMBASE`, *or* `INVOICE.ITEMSUMENTERPRISE` *depending on the user's choice of Currency Type*
[Invoiced Net - Transaction]	`INVOICE.ITEMSUMCURRENCY`
[Invoiced Net - Company]	`INVOICE.ITEMSUMBASE`
[Invoiced Net - Enterprise]	`INVOICE.ITEMSUMENTERPRISE`
[Tax]	`INVOICE.VATCURRENCY`, `INVOICE.VATBASE`, *or* `INVOICE.VATENTERPRISE` *depending on the user's choice of Currency Type*
[Tax - Transaction]	`INVOICE.VATCURRENCY`

[Tax - Company] INVOICE.VATBASE
[Tax - Enterprise] INVOICE.VATENTERPRISE

R Notice that invoice objects are not consistently available in all of the four currency types. The reason is that only the invoiced amount is available in the currency of the customer.

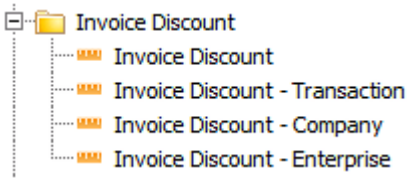
All the charge values related to invoiced amounts are available for reporting on, including their tax amounts. Note that charge amounts are not available in the currency of the customer as this is not available in Maconomy either.



[Charge 1]	INVOICE.CHARGE1CURRENCY, INVOICE.CHARGE1BASE, *or* INVOICE.CHARGE1ENTERPRISE *depending on the user's choice of Currency Type*
[Charge 1 - Transaction]	INVOICE.CHARGE1CURRENCY
[Charge 1 - Transaction]	INVOICE.CHARGE1BASE
[Charge 1 - Transaction]	INVOICE.CHARGE1ENTERPRISE
[Charge 2]	INVOICE.CHARGE2CURRENCY, INVOICE.CHARGE2BASE, *or* INVOICE.CHARGE2ENTERPRISE *depending on the user's choice of Currency Type*
[Charge 2 - Transaction]	INVOICE.CHARGE2CURRENCY
[Charge 2 - Transaction]	INVOICE.CHARGE2BASE
[Charge 2 - Transaction]	INVOICE.CHARGE2ENTERPRISE
[Charge 3]	INVOICE.CHARGE3CURRENCY, INVOICE.CHARGE3BASE, *or* INVOICE.CHARGE3ENTERPRISE *depending on the user's choice of Currency Type*
[Charge 3 - Transaction]	INVOICE.CHARGE3CURRENCY
[Charge 3 - Transaction]	INVOICE.CHARGE3BASE
[Charge 3 - Transaction]	INVOICE.CHARGE3ENTERPRISE
[Charge 4]	INVOICE.CHARGE4CURRENCY, INVOICE.CHARGE4BASE, *or* INVOICE.CHARGE4ENTERPRISE *depending on the user's choice of Currency Type*
[Charge 4 - Transaction]	INVOICE.CHARGE4CURRENCY
[Charge 4 - Transaction]	INVOICE.CHARGE4BASE
[Charge 4 - Transaction]	INVOICE.CHARGE4ENTERPRISE

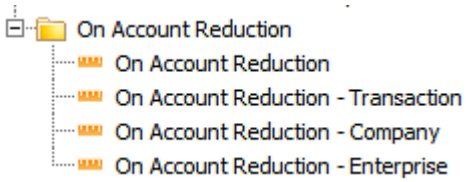
[Charge 1 Tax]	INVOICE.CHARGE1VATCURRENCY, INVOICE.CHARGE1VATBASE, *or* INVOICE.CHARGE1VATENTERPRISE *depending on the user's choice of Currency Type*
[Charge 1 Tax - Transaction]	INVOICE.CHARGE1VATCURRENCY
[Charge 1 Tax - Transaction]	INVOICE.CHARGE1VATBASE
[Charge 1 Tax - Transaction]	INVOICE.CHARGE1VATENTERPRISE
[Charge 2 Tax]	INVOICE.CHARGE2VATCURRENCY, INVOICE.CHARGE2VATBASE, *or* INVOICE.CHARGE2VATENTERPRISE *depending on the user's choice of Currency Type*
[Charge 2 Tax - Transaction]	INVOICE.CHARGE2VATCURRENCY
[Charge 2 Tax - Transaction]	INVOICE.CHARGE2VATBASE
[Charge 2 Tax - Transaction]	INVOICE.CHARGE2VATENTERPRISE
[Charge 3 Tax]	INVOICE.CHARGE3VATCURRENCY, INVOICE.CHARGE3VATBASE, *or* INVOICE.CHARGE3VATENTERPRISE *depending on the user's choice of Currency Type*
[Charge 3 Tax - Transaction]	INVOICE.CHARGE3VATCURRENCY
[Charge 3 Tax - Transaction]	INVOICE.CHARGE3VATBASE
[Charge 3 Tax - Transaction]	INVOICE.CHARGE3VATENTERPRISE
[Charge 4 Tax]	INVOICE.CHARGE4VATCURRENCY, INVOICE.CHARGE4VATBASE, *or* INVOICE.CHARGE4VATENTERPRISE *depending on the user's choice of Currency Type*
[Charge 4 Tax - Transaction]	INVOICE.CHARGE4VATCURRENCY
[Charge 4 Tax - Transaction]	INVOICE.CHARGE4VATBASE
[Charge 4 Tax - Transaction]	INVOICE.CHARGE4VATENTERPRISE

Amounts for invoice discount are available in the three currency types: Transaction, Company and Enterprise. The invoice discount amount is not available in the currency of the customer as it is not either in Maconomy.



[Invoice Discount]	INVOICE.INVOICEDISCOUNTCURRENCY, INVOICE.INVOICEDISCOUNTBASE, *or* INVOICE.INVOICEDISCOUNTENTERPRISE *depending on the user's choice of Currency Type*
[Invoice Discount - Transaction]	INVOICE.INVOICEDISCOUNTCURRENCY
[Invoice Discount - Company]	INVOICE.INVOICEDISCOUNTBASE
[Invoice Discount - Enterprise]	INVOICE.INVOICEDISCOUNTENTERPRISE

Amounts for on account reduction are available in the three currency types: Transaction, Company and Enterprise. The invoice discount amount is not available in the currency of the customer as it is not either in Maconomy.



[On Account Reduction]	INVOICE.ONACCOUNTITEMSUMCURRENCY, INVOICE.ONACCOUNTITEMSUMBASE, *or* INVOICE.ONACCOUNTITEMSUMENTERPRISE *depending on the user's choice of Currency Type*
[On Account Reduction - Transaction]	INVOICE.ONACCOUNTITEMSUMCURRENCY
[On Account Reduction - Company]	INVOICE.ONACCOUNTITEMSUMBASE
[On Account Reduction - Enterprise]	INVOICE.ONACCOUNTITEMSUMENTERPRISE

Chapter 16

AP Aging Universe

The AP Aging universe provides objects for reporting on outstanding vendor balances. The balances are the amounts that the company has not paid to vendors concerning vendor invoices or similar. The outstanding balances can be aged according to the aging principles set up in Maconomy and thereby distinguish the outstandings on how old they are. It is possible to report on the associated vendor invoice journals or expense sheets, as well as the related payment information. Furthermore, it is possible to report on figures and status of electronic and manual payments including checks.

The measure values can be grouped and restricted by a variety of dimensions including company, vendor, employee, dates, and the ten standard dimensions.

16.1 Prerequisites

In order to fully benefit from the AP Aging universe, Maconomy should be set up such that:

- The system parameter **Use Calendar Day** is enabled and the calendar spans all necessary dates.
- In order to report on the aging of outstandings, an aging principle must be set up.

16.2 Maconomy Workflows

The workflows related to the AP Aging universe has many similarities with the workflows related to the AR Aging universe. The workflows related to the AP Aging are (data wise) centered around records in the table `VENDORENTRY`, where some represent **Vendor Invoices** and some represent corresponding payments. As for AR, the two types of entries are engaged in a reconciliation process that match them out. In AP, this results in records in the table `VENDORRECONCILIATION` table as well as records in the `VENDORRECONCILIATIONJOURNAL` table.

16.2.1 Vendor Invoices, Payment, and Reconciliation

Consider the following:

1. In the windows **Vendor Invoices**, create a new journal.
2. Add a single line in the lower panel with an amount of $5000.
3. Submit and post the journal.

   This creates a new `VENDORENTRY` record representing the vendor invoice just created.
4. In the window **Change Payment Selection by Vendor**, locate the vendor.
5. In the lower panel, enter an amount to be paid. For this scenario, enter the same amount as invoiced.
6. Approve the payment.

   This provides another `VENDORENTRY` representing the payment.
7. In the window **A/P Open Entry Reconciliation**, locate the vendor entry representing the vendor invoice. The corresponding payment will appear in the lower panel.
8. Reconcile the entries fully.

   This will create a record in the table `VENDORRECONCILIATIONJOURNAL` and one in the table `VENDORRECONCILIATION`.

16.2.2 Payment with Checks

In this scenario we shall make the payment by check. We shall also vary the scenario by issuing two vendor invoices while still having one payment.

1. In the windows **Vendor Invoices**, create a new journal.
2. Add two lines in the lower panel: one with an amount of $2000 and one of $1000. The two lines should have the same vendor but different invoice numbers.
3. Make sure that the payment mode is set to one of the “Check” payment types available.
4. Submit and post the journal.

   This provides two new records in the `VENDORENTRY` table.
5. In the window **Change Payment Selection by Vendor**, locate the vendor.
6. In the lower panel, enter an amount to be paid. For this scenario, enter the same amount as invoiced.

7. Approve the payment.

   This provides another `VENDORENTRY` *representing the payment. Furthermore, it creates a record in the table* `OUTPUTDATALINE` *which represents payments via the bank. Notice, that this was also the case in the previous scenario but just not relevant to explain. The output data line is important in relation to reporting on the status of checks as such status information is available on that type of record and not on the vendor entry.*

8. In the window **A/P Open Entry Reconciliation** find the vendor entry representing the vendor invoice. The corresponding payment will appear in the lower panel.

9. Reconcile the entries fully.

   As in the previous scenario, it will create two records in the table `VENDORRECONCILIATION` *and one in the table* `VENDORRECONCILIATIONJOURNAL`*. It will also result in two records in the table* `OUTPUTDATARECONCILIATION`*. The records in this table can be used for relating the payment (record in* `OUTPUTDATALINE`*) to the vendor invoices (records in* `VENDORENTRY`*.*

16.3 Business Layer

The universe provides measures for reporting on different kinds of registrations related to **Accounts Payable**. There are objects both for building AP Aging reports and also for reporting on the individual transactional-near debit and credit amounts, tax amounts, and associated dimensions.

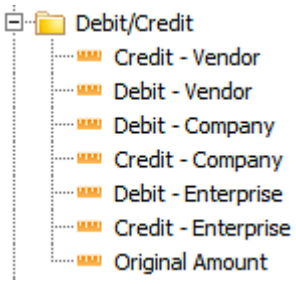
It is only the balance objects (or objects using these in their calculation like the period objects) that really utilize the vendor reconciliations and vendor reconciliation journals that are related to vendor entries. Other measures basically calculate entirely from fields on the vendor entries.

In the following sub-sections, we shall go through each of the main folders and explain the measure objects contained.

16.3.1 Debit/Credit

Objects for reporting on the individual debit and credit figures of vendor entries are located in the [`Debit/Credit`] folder.

These objects only concern figures stored on the vendor entries. No reconciliations are taken into account so it is only the pure transactional figures that are provided. There are debit and credit objects in the three currency types *Company*, *Vendor*, and *Enterprise*. Furthermore, there is an object for reporting on the amount posted in the original currency of the transaction.



[Debit - Vendor]	[VENDORENTRY.DEBITCURRENCY]
[Credit - Vendor]	[VENDORENTRY.CREDITCURRENCY]
[Debit - Company]	[VENDORENTRY.DEBITBASE]
[Credit - Company]	[VENDORENTRY.CREDITBASE]
[Debit - Enterprise]	[VENDORENTRY.DEBITENTERPRISE]
[Credit - Enterprise]	[VENDORENTRY.CREDITENTERPRISE]
[Original Amount]	[VENDORENTRY.ORIGINALAMOUNT]

16.3.2 Remainders and Payments

Objects for reporting on the remainders and payments to vendors, are located in the [Remainder/Being Paid] folder. To report on the remainders and amounts being paid—as represented on vendor entries—we utilize dedicated database fields directly.

R Notice, that the REMAINDER... and AMOUNTBEINGPAID... fields on VENDORENTRY do not take any statement date into account. Hence, they are not suitable for reporting historically on outstanding and paid amounts. For outstanding amounts, use the Balance objects. For reporting on vendor payments—like checks or electronic payments—use objects in the Bank universe or in the Vendor Payment universe; see Chapter 11 and Chapter 17.

There are three objects for remainders and three for amounts being paid; one in each of the three currency types.

[Remainder - Vendor] [VENDORENTRY.REMAINDERCURRENCY]
[Remainder - Company] [VENDORENTRY.REMAINDERCURRENCY]
[Remainder - Enterprise] [VENDORENTRY.REMAINDERENTERPRISE]
[Amount Being Paid - Vendor] [VENDORENTRY.AMOUNTBEINGPAID]
[Amount Being Paid - Company] [VENDORENTRY.AMOUNTBEINGPAIDCOMPANY]
[Amount Being Paid Enterprise] [VENDORENTRY.AMOUNTBEINGPAID]

16.3.3 Tax

Objects for reporting on the tax amounts of vendor entries, are located in the [Tax] folder. To report on tax amounts, we use the dedicated TAX... fields on VENDORENTRY.

There are three tax objects; one for each of the three currency types.

[Tax - Vendor] [VENDORENTRY.TAXCURRENCY]
[Tax - Company] [VENDORENTRY.TAXBASE]
[Tax - Enterprise] [VENDORENTRY.TAXENTERPRISE]
[Amount Being Paid - Vendor] [VENDORENTRY.AMOUNTBEINGPAID]

16.3.4 Balances

Objects for reporting on the balance of vendors, are located in the [Balances] folder. By a *balance* we understand the summation of the balances for each vendor entry (*debit–credit*), subtracting any possible reconciled amounts. The reconciled amounts are either stored on records in the table VENDORRECONCILIATION or VENDORRECONCILIATIONJOURNAL depending on whether the vendor entry in question appeared in the lower or upper panel of the window **Vendor Open Entry Reconciliation**, when the reconciliation was performed.

R Depending on whether the balance on the vendor entry is negative, we either subtract or add the reconciled amount.

Balances are always calculated according to a *statement date*. This date applies as a restriction on both vendor entries and the reconciling entries in VENDORRECONCILIATION and VENDORRECONCILIATIONJOURNAL:

- Only vendor entries that *exist* on the statement date, are included. We can choose either to use the ENTRYDATE or the DUEDATE of the vendor entry. E.g. using the ENTRYDATE, we include the vendor entry if the entry date is before or equal to the statement date. If it is after, we do not consider the entry to exist yet and thereby exclude it.
- Only vendor reconciliations and vendor reconciliation journals that have been applied on the statement date, are included. Reconciliations actually have two dates as well:
  - the RECONCILIATIONDATE which is the date the reconciliation took place.
  - the ENTRYRECONCILIATIONDATE which is a date the user can enter to mean the date from where it should count.

  In Maconomy and BPM, we do not consider a vendor reconciliation to have fully occurred until both dates have passed. This means that we only include vendor reconciliations and vendor reconciliation journals for which both the RECONCILIATIONDATE and ENTRYRECONCILIATIONDATE are before or equal the statement date. If they are after the statement date, we do not consider the reconciliation to have occurred yet.

R The VENDORENTRY table contains fields that state the *remainder* of the entry. However, these fields do not take any statement date into account. They state what the remainder balance is on the entry regardlessly of when we report on it. E.g. if we have a vendor entry with an entry date of January 1^{st} and fully reconciled a month later. The remainder will be zero even though we report between the creation of the entry and before the reconciliation. The old **Maconomy Analyzer** report for AP Aging would behave in the same way; i.e. not work historically concerning when entries exist and reconciliations are done.

In Maconomy, reconciliations can also be removed. This is done if it was an error to reconcile in the first place. When a reconciliation is removed, an additional entry in the table `VENDORRECONCILIATIONJOURNAL` is created. The original entry in that table is marked with `REMOVEDRECONCILIATION`=1. On the corresponding entries in the table `VENDORRECONCILIATION`, the field `REMOVEDRECONCILIATIONDATE` is assigned a date. As we typically do not want to include reconciliations that are removed, we want only to include the vendor reconciliation journals with `REMOVEDRECONCILIATION`=0 and the vendor reconciliations with `REMOVEDRECONCILIATIONDATE` = `NULL`.

When a reconciliation is removed in Maconomy it means that the reconciliation should never have been done. Therefore it does not make sense to report historically on when a reconciliation was removed, and BPM does not support this.

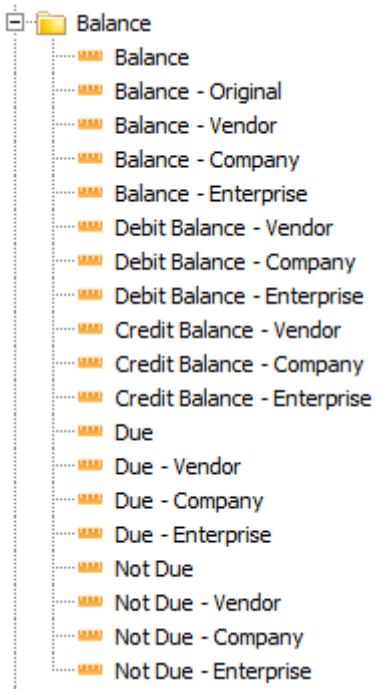
When using the objects in the [`Balance`] folder, an aging principle needs to be applied. This aging principle defines whether the statement date is using the entry date or the due date of vendor entries. It also defines many other things but this is the most important for the balance objects. For more information about the aging principles, see Section 16.3.5.

The [Balances] folder contains three kinds of objects:

- The Balance objects.
- The Due objects.
- The Not Due objects.

The balance objects. There are a total of four objects. One for each of the three currency types: *Vendor*, *Company* and *Enterprise*, and one for displaying in the original currency of the transaction. There is also an object [Balance] which—depending on the choice of currency type—issues one of the three currency type specific balance objects. The balance object in original currency is a special version of the balance objects. In the Maconomy database we do not have the reconciled amounts available in the case where the currency of the transaction is not the currency of the vendor, company or enterprise. Therefore we estimate the currency exchange rate between the vendor's currency and the currency of the transaction by calculating the quotient between the balance in the vendor's currency and the debit minus credit of the transaction in the vendors currency. This quotient we multiply with the original amount.

The due/not due objects are special versions of the balance objects, where the aging principle and statement date are used for determining the balance that is due (i.e. over statement date) and not due, respectively. The objects [Due] and [Not Due] prompt the user for choosing a currency type.



[Balance]	[Balance - Company], [Balance - Customer] *or* [Balance - Enterprise] *depending on the choice of currency type*
[Balance - Original]	[Balance - Vendor] *if* CUSTOMERENTRY.ORIGNALCURRENCY=ENTRYCURRENCY

[Balance - Company]
if CUSTOMERENTRY.ORIGNALCURRENCY=COMPANYCURRENCY

[Balance - Enterprise]
if CUSTOMERENTRY.ORIGNALCURRENCY=BASECURRENCY

$$\frac{\texttt{[[Balance - Vendor]]} * \texttt{CUSTOMERENTRY.ORIGINALAMOUNT}}{\texttt{CUSTOMERENTRY.CREDITCURRENCY} - \texttt{CUSTOMERENTRY.DEBITCURRENCY}}$$

if CUSTOMERENTRY.CREDITCURRENCY
$-$CUSTOMERENTRY.DEBITCURRENCY $\neq 0$

0 *otherwise*

[Balance - Vendor]

$$\begin{aligned}
&\texttt{VENDORENTRY.CREDITCURRENCY} \\
-\ &\texttt{VENDORENTRY.DEBITCURRENCY} \\
-\ &\textstyle\sum \texttt{VENDORRECONCILIATION.RECONCILIATIONSTANDARD} \\
-\ &\textstyle\sum \texttt{VENDORRECONCILIATIONJOURNAL.RECONCILEDENTRYSTANDARD} \\
&\textit{if } \texttt{CUSTOMERENTRY.CREDITCURRENCY} \\
&\quad -\texttt{CUSTOMERENTRY.DEBITCURRENCY} > 0 \\
&\textit{or } \texttt{CUSTOMERENTRY.CREDITBASE} \\
&\quad -\texttt{CUSTOMERENTRY.DEBITBASE} > 0 \\
&\textit{or } \texttt{CUSTOMERENTRY.CREDITENTERPRISE} \\
&\quad -\texttt{CUSTOMERENTRY.DEBITENTERPRISE} > 0
\end{aligned}$$

$$\begin{aligned}
&\texttt{VENDORENTRY.CREDITCURRENCY} \\
-\ &\texttt{VENDORENTRY.DEBITCURRENCY} \\
+\ &\textstyle\sum \texttt{VENDORRECONCILIATION.RECONCILIATIONSTANDARD} \\
+\ &\textstyle\sum \texttt{VENDORRECONCILIATIONJOURNAL.RECONCILEDENTRYSTANDARD} \\
&\textit{otherwise}
\end{aligned}$$

[Balance - Company]

$$\begin{aligned}
&\texttt{VENDORENTRY.CREDITBASE} \\
-\ &\texttt{VENDORENTRY.DEBITBASE} \\
-\ &\textstyle\sum \texttt{VENDORRECONCILIATION.RECONCILIATIONBASE} \\
-\ &\textstyle\sum \texttt{VENDORRECONCILIATIONJOURNAL.RECONCILEDENTRYBASE} \\
&\textit{if } \texttt{CUSTOMERENTRY.CREDITCURRENCY} \\
&\quad -\texttt{CUSTOMERENTRY.DEBITCURRENCY} > 0 \\
&\textit{or } \texttt{CUSTOMERENTRY.CREDITBASE} \\
&\quad -\texttt{CUSTOMERENTRY.DEBITBASE} > 0 \\
&\textit{or } \texttt{CUSTOMERENTRY.CREDITENTERPRISE} \\
&\quad -\texttt{CUSTOMERENTRY.DEBITENTERPRISE} > 0
\end{aligned}$$

VENDORENTRY.CREDITBASE
$-$ VENDORENTRY.DEBITBASE
$+$ $\sum$ VENDORRECONCILIATION.RECONCILIATIONBASE
$+$ $\sum$ VENDORRECONCILIATIONJOURNAL.RECONCILEDENTRYBASE
otherwise

[Balance - Enterprise]

VENDORENTRY.CREDITENTERPRISE
$-$ VENDORENTRY.DEBITENTERPRISE
$-$ $\sum$ VENDORRECONCILIATION.RECONCILIATIONENTERPRISE
$-$ $\sum$ VENDORRECONCILIATIONJOURNAL.RECONCILEDENTRYENTERPRISE
if CUSTOMERENTRY.CREDITCURRENCY
$-$CUSTOMERENTRY.DEBITCURRENCY > 0
or CUSTOMERENTRY.CREDITBASE
$-$CUSTOMERENTRY.DEBITBASE > 0
or CUSTOMERENTRY.CREDITENTERPRISE
$-$CUSTOMERENTRY.DEBITENTERPRISE > 0

VENDORENTRY.CREDITENTERPRISE
$-$ VENDORENTRY.DEBITENTERPRISE
$+$ $\sum$ VENDORRECONCILIATION.RECONCILIATIONENTEPRISE
$+$ $\sum$ VENDORRECONCILIATIONJOURNAL.RECONCILEDENTRYENTERPRISE
otherwise

[Debit Balance - Vendor]

VENDORENTRY.DEBITCURRENCY
$-$ $\sum$ VENDORRECONCILIATION.RECONCILIATIONSTANDARD
$-$ $\sum$ VENDORRECONCILIATIONJOURNAL.RECONCILEDENTRYSTANDARD
if CUSTOMERENTRY.DEBITCURRENCY > 0
or CUSTOMERENTRY.DEBITBASE > 0
or CUSTOMERENTRY.DEBITENTERPRISE > 0

0 *otherwise*

[Debit Balance - Company]

VENDORENTRY.DEBITBASE
$-$ $\sum$ VENDORRECONCILIATION.RECONCILIATIONBASE
$-$ $\sum$ VENDORRECONCILIATIONJOURNAL.RECONCILEDENTRYBASE
if CUSTOMERENTRY.DEBITCURRENCY > 0
or CUSTOMERENTRY.DEBITBASE > 0
or CUSTOMERENTRY.DEBITENTERPRISE > 0

0 *otherwise*

[Debit Balance - Enterprise]

	VENDORENTRY.DEBITENTERPRISE $-$ $\sum$ VENDORRECONCILIATION.RECONCILIATIONENTERPRISE $-$ $\sum$ VENDORRECONCILIATIONJOURNAL.RECONCILEDENTRYENTERPRISE *if* CUSTOMERENTRY.DEBITCURRENCY > 0 *or* CUSTOMERENTRY.DEBITBASE > 0 *or* CUSTOMERENTRY.DEBITENTERPRISE > 0 0 *otherwise*
[Credit Balance - Vendor]	VENDORENTRY.CREDITCURRENCY $-$ $\sum$ VENDORRECONCILIATION.RECONCILIATIONSTANDARD $-$ $\sum$ VENDORRECONCILIATIONJOURNAL.RECONCILEDENTRYSTANDARD *if* CUSTOMERENTRY.CREDITCURRENCY > 0 *or* CUSTOMERENTRY.CREDITBASE > 0 *or* CUSTOMERENTRY.CREDITENTERPRISE > 0 0 *otherwise*
[Credit Balance - Company]	VENDORENTRY.CREDITBASE $-$ $\sum$ VENDORRECONCILIATION.RECONCILIATIONBASE $-$ $\sum$ VENDORRECONCILIATIONJOURNAL.RECONCILEDENTRYBASE *if* CUSTOMERENTRY.CREDITCURRENCY > 0 *or* CUSTOMERENTRY.CREDITBASE > 0 *or* CUSTOMERENTRY.CREDITENTERPRISE > 0 0 *otherwise*
[Credit Balance - Enterprise]	VENDORENTRY.CREDITENTERPRISE $-$ $\sum$ VENDORRECONCILIATION.RECONCILIATIONENTERPRISE $-$ $\sum$ VENDORRECONCILIATIONJOURNAL.RECONCILEDENTRYENTERPRISE *if* CUSTOMERENTRY.CREDITCURRENCY > 0 *or* CUSTOMERENTRY.CREDITBASE > 0 *or* CUSTOMERENTRY.CREDITENTERPRISE > 0 0 *otherwise*
[Due - Vendor]	[Balance - Vendor] *if* [Aging Units Due] > 0
[Due - Company]	[Balance - Company] *if* [Aging Units Due] > 0
[Due - Enterprise]	[Balance - Enterprise] *if* [Aging Units Due] > 0
[Not Due - Vendor]	[Balance - Vendor] *if* [Aging Units Due] ≤ 0

[Not Due - Company] [Balance - Company] *if* [Aging Units Due] ≤ 0
[Not Due - Enterprise] [Balance - Enterprise] *if* [Aging Units Due] ≤ 0

16.3.5 Aging

The objects for reporting on aged vendor balances, are located in the folder [Aging]. An aged amount is a vendor entry balance that is categorized to belong to one of several aging periods. The aging periods are defined by an **Aging Principle** in Maconomy. Each aging principle states whether to use the entry date or the due date, and it states whether periods are defined by days or months. Furthermore, an aging principle contains a collection of up to 10 periods. Each period states a *from* and a *to* limit but the period can also be open. E.g. we can define a period from 30 days and beyond. Each period is assigned a text; e.g. 30+ or 0-29.

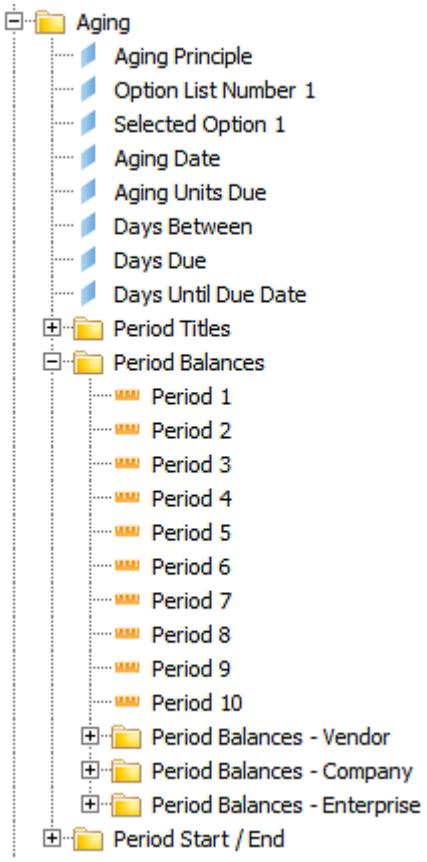
In BPM Reporting, we need to determine the limits of the aging periods in order to find out which period each vendor entry (or reconciliation) belongs to. To do so, we first calculate the lower and upper limit of each period. This is represented as an integer. If the aging principle is day-based the integers count the number of days. If it is month-based the integers count the number in months. If the interval is open, we use the value 9999 or -9999; depending on whether it is an upper limit or a lower limit; and depending on whether the period is looking ahead or backwards.

When we have all the limits, we can determine whether a balance belongs to a period restricted by a lower and an upper limit. The result is the calculation of 10 measure objects [Period 1]-[Period 10].

There are 10 [`Period`] objects that will prompt the user for choosing a currency type: *Vendor*, *Company*, or *Enterprise*. Furthermore, there are 10 similar measure objects for each of the three currency types. Below, we have stated the calculations of these measure objects assuming the currency of the company.

The folder `Period Titles` contains 10 dimension objects stating the chosen title for each of the 10 periods. These are convenient to use as headers in report tables.

The folder `Period Start/End` contains the dimension objects that calculate the lower and upper period limits. These can be convenient if defining new kinds of period objects. Furthermore, we have a couple of dimensions for determining the aging principle (the objects [`Aging Principle`], [`Option List Number 1`], and [`Selected Option 1`]). The object [`Aging Date`] states either the entry date or the due date, depending on what is set for the aging principle. The object [`Aging Units Due`] is a helper-object that states the number of units (days or months) that an entry is due. A positive number means overdue. The object [`Days Between`] is another helper-object that states the number of days between the entry and the statement date.



Object	Database Field
[Period 1]	[Balance - Company] if [Aging Units Due] ∈ [[Period 1 Start]; [Period 1 End]]
[Period 2]	[Balance - Company] if [Aging Units Due] ∈ [[Period 2 Start]; [Period 2 End]]
[Period 3]	[Balance - Company] if [Aging Units Due] ∈ [[Period 3 Start]; [Period 3 End]]
[Period 4]	[Balance - Company] if [Aging Units Due] ∈ [[Period 4 Start]; [Period 4 End]]
[Period 5]	[Balance - Company] if [Aging Units Due] ∈ [[Period 5 Start]; [Period 5 End]]
[Period 6]	[Balance - Company] if [Aging Units Due] ∈ [[Period 6 Start]; [Period 6 End]]
[Period 7]	[Balance - Company] if [Aging Units Due] ∈ [[Period 7 Start]; [Period 7 End]]
[Period 8]	[Balance - Company] if [Aging Units Due] ∈ [[Period 8 Start]; [Period 8 End]]
[Period 9]	[Balance - Company] if [Aging Units Due] ∈ [[Period 9 Start]; [Period 9 End]]
[Period 10]	[Balance - Company] if [Aging Units Due] ∈ [[Period 10 Start]; [Period 10 End]]

Similar objects are present for vendor currency and enterprise currency.

16.4 Reporting Examples

The reporting capabilities of the AP Aging universe is almost identical to the capabilities of the AP Aging universe. Therefore, the examples from Section 14.4 can easily be translated to AP. Below, we shall just give one of these similar examples.

16.4.1 Vendor Balances

By a **Vendor Balance**, we understand the amount that in total has been invoiced by vendors but not yet paid and reconciled. This is really the outstanding balance on vendors. In context of vendor balances, we do not consider payments to count until they are reconciled. If a payment is engaged in a partial reconciliation, only the reconciled amount is included in the calculation of the outstanding amount. I.e., the similar principles that apply for **Customer Reconciliation** and **Customer Outstandings** also applies to **Vendor Reconciliation** and **Vendor Outstanding**

Example 16.1 (Reporting on vendor balances) *In this first example, we shall create a report that outlines the outstanding balance of vendors.*

1. *Create a new WebIntelligence document and select the universe* AP Aging
2. *Add the following objects for the query:*

   [Currency] *(folder [Currency])*

   [Vendor No.] *(folder [Company Vendor])*

   [Vendor Name] *(folder [Company Vendor])*

   [Balance] *(folder [Balance])*
3. *Run the report*

The report will prompt the user for selecting in which currency, the outstanding amount should be shown. The report shows a table of four columns: Currency, Vendor Number, Vendor Name, and the outstanding amount for the vendor displayed on the row. For convenience, you may drag the currency column up above the table to form a section on currencies.

This report will show all the outstandings of vendors. It uses whatever data is in the Maconomy database, but it is not working historically with regards to dates. If we have an invoice that was created before the date the report is run but paid and reconciled with entry reconciliation date in the future (which is possible), the report will still pick up that reconciliation. See the reporting examples of the AR Aging universe for how to consider a statement date; e.g. Section 14.4. ■

Chapter 17

Vendor Payment Universe

The Vendor Payment universe provides objects for reporting on vendor payments. It is possible to report on paid amounts, remainders, withholding tax, and cash discounts.

The payment amounts can be grouped and restricted by a variety of dimensions including payment dates, standard dimensions, company and vendor. Furthermore, it is possible to report on various transactional data related to the payments.

17.1 Prerequisites

In order to fully benefit from the Vendor Payment universe, Maconomy should be set up such that:

- The system parameter **Use Calendar Day** is enabled and the calendar spans all necessary dates.

Chapter 18

Reporting Structure Universe

The Reporting Structure universe provides objects for reporting on all the reporting structures set up in Maconomy. The purpose of the universe is to make it possible to utilize reporting structures for structuring and filtering data coming from other universes. Thus, it is possible to group, order and filter data on a chosen dimension. E.g. a hierarchy of transaction types can be defined as a reporting structure, and the groups of that hierarchy can then be combined with financial data.

The universe contains objects for reporting on the twelve possible grouping levels and the sorting on each level. It is also possible to report on the sign values that can be set on each grouping level. Furthermore, the universe contains the necessary objects for selecting and filtering reporting structures, like option list and selected option, the type of dimension the reporting structure is made on, the date interval, etc.

18.1 More than grouping

Reporting structures are actually much more powerful than just a feature for grouping dimensions. In fact, their purpose is three-fold:

Grouping: The hierarchical groups of the reporting structure makes it possible to organize and display data grouped as the reporting structure defines. E.g. we can display groups and sub-groups of activities and combine these dimensions with job cost registration measures like revenue recognized.

Ordering: The order of the reporting structure can be used for ordering data. This way, the order does not have to follow the lexical order of the chosen dimension. E.g. finance figures summarized by account number, can follow the order in which we have stated the accounts in the reporting structure; instead of increasing order of the account number.

Filtering: A reporting structure does not have to include all the dimension values

available. We could choose to only include a subset. Thereby, we can control which figures to display by which dimensions to include. E.g. we could define a reporting structure that only includes certain transaction types. Then when combining that reporting structure with job cost data, we also filter what job cost data to include; namely the data done to the included transaction types.

18.2 Dimensions with or without data

When we combine data fetched from a query to the Reporting Structure universe with data fetched from another universe, we can choose between two strategies for display:

- *Only display groups for which there are data.* This means that we display the data from one query and associate reporting structure groups. However, reporting structure lines and groups for which there are no matching dimension value in the data query, are left out.
- *Display groups despite whether there are data or not.* This means that we display the reporting structure lines (and associated dimension) and associate the data from the other query if they exist. If not, blank or zero values are displayed for the dimension value on the reporting structure line. I.e. if data does not exist for a given dimension, we still display the reporting structure group or dimension.

Consider the standard finance reports of BPM Reporting. Most of them offer the ability to display zero lines. This means that the reports list accounts and account groups even though no transactions exist in the selected fiscal period. This functionality corresponds to *Display groups despite whether there are data or not.* When we choose to *not* display zero lines, it corresponds to *Only display groups for which there are data.* The Reporting Structure universe offers both facilities.

18.3 Identifying Reporting Structures

Each reporting structure has a unique name that is chosen on the creation of the reporting structure. In principle, a report could prompt the user for selecting a reporting structure by prompting the user for choosing a reporting structure name. However, this is not convenient for at least two reasons:

- Reporting structure names are rather internal setup data and may not be known to ordinary report users.
- Reporting structure names are not localized so they may not make sense if running on other locales. Imagine what a French report user (in a multi-country company) would answer if having to select a reporting structure where these were given German names because the main company is German.

It is recommended to assume internal references instead and not prompting the user. For this purpose, a reporting structure can be assigned an option list and a selected

option.

Furthermore, reporting structures are identified by the type of their dimension (e.g. Account, Transaction Type or Activity), and the date period for which they are valid. It is thus possible to create multiple reporting structures on the same dimension, referencing the same option list and value, as long as the reporting structures have distinct non-overlapping date intervals.

18.4 Prerequisites

There are no specific prerequisites, except that the necessary reporting structures should be set up in Maconomy.

18.5 Maconomy Workflows

In the following sub-sections, we shall look at some simple Maconomy workflows that lead to the creation of specific reporting structures, and we then explain how they are represented in the database and what reporting capabilities they offer.

18.5.1 Account Reporting Structure

1. In the window **Option Lists**, create a new option list with the name: `Standard Reporting Structures`.
2. Add an option `BPM Finance` to the list.
3. In the window **Reporting Structures**, create a new reporting structure on the dimension **Account**.
4. Run the action `Add Missing`. This will add a line in the reporting structure for each global account in Maconomy.
5. Repeatingly, add grouping lines and indent the account lines below to form a suitable hierarchy (see Figure 18.1 that displays part of such a hierarchy.)

18.5.2 Group Signs

Each line in a reporting structure has an `Invert Sign` field. The purpose of this field is to state whether figures related to the dimensions in that group should have their sign flipped or not. The field is general purpose and does not change figures in Maconomy but is purely there so that we can utilize it when we display figures in BPM reports.

The `Invert Sign` field can be used for changing to a normal display of finance data, as opposed to *bookkeeping-style*.

Say that we have a couple of revenue accounts and a couple of cost accounts. When posting a sales, it is a credit on the sales account but we may want to display sales

Reporting Structure Lines | Add Selected Accounts | Add Missing | Move Lines

Show

	Classification	Account No.	Description	Invert Sign	Selected for Moving
1	**Net Profit**		**Net Profit**	☐	☐
2	**EBITDA**		**EBITDA**	☐	☐
3	**Gross Profit**		**Gross Profit**	☐	☐
4	**Revenue**		**Revenue**	☑	☐
5	1110	1110	Sales, Taxable	☑	☐
6	1120	1120	Sales, Export	☑	☐
7	1130	1130	Sales, Nontaxable	☑	☐
8	1140	1140	Sales, Accrual	☑	☐
9	1145	1145	Open Sales, Accrual	☑	☐
10	1150	1150	Sales, Furniture	☑	☐
11	1155	1155	Sales, Misc.	☑	☐
12	**Job Revenue**		**Job Revenue**	☑	☐
23	**Intercompany Revenue**		**Intercompany Revenue**	☑	☐
28	**Unbilled Revenue**		**Unbilled Revenue**	☑	☐
38	**Cost**		**Cost**	☐	☐
39	**Cost of Revenue**		**Cost of Revenue**	☐	☐
40	1710	1710	Cost of Sales	☐	☐
41	1715	1715	Cost of Sales, Accrual	☐	☐
42	1716	1716	Accrued Cost of Sales, Accrual	☐	☐
43	1720	1720	Cost of Sales, Intercompany	☐	☐

Figure 18.1: Account Reporting Structure

accounts in the revenue account group using positive sign. This can be achieved by "ticking" the `Invert Sign` field on the lines of that group of accounts in the reporting structure.[1] This is done for the revenue accounts in the reporting structure shown in Figure 18.1. The values on the cost accounts should still display as positive (they are debits already) and we know that they are costs, so naturally we should not flip their sign (as also shown on Figure 18.1. Usually, we group revenue and cost under a group that we could call *Gross Profit*. The summation of revenue and cost is a summation of data as they appear in the database. The sign flipping is just a matter of display in BPM reports; if utilized. However, we wish to display gross profit as positive if revenue less cost is positive. This means that we need to flip the sign of the *Gross Profit* group.

Table 18.1 shows how the sign flipping works in general and how it can be utilized to control the sign of figures on the different account grouping levels.

Assume that `1010` and `1020` are sales accounts, and `2210` and `2230` are cost accounts. `1010` and 1020 are grouped under `Revenue`. `2010` and `2230` are grouped under `Cost`. Both groups are under `Gross Profit`. With some debited amounts and some credited amounts, it could look something like what is shown in Table 18.1:

[1] Here we assume that the respective finance report indeed utilizes the `Invert Sign` field.

Accounts	Database Debit	Credit	Balance	Invert Sign	Display Balance
Gross Profit				Yes	
Revenue				Yes	
1010		2500	-2500	Yes	2500
1020		7000	-7000	Yes	7000
Revenue Total:		**9500**	**-9500**	(Yes)	**9500**
Cost					
2210	1600		1600	No	1600
2230	4000		4000	No	4000
Cost Total:	**5600**		**5600**	(No)	**5600**
Gross Profit Total:	**5600**	**9500**	**-3900**	(Yes)	**3900**

Table 18.1: Invert sign and calculation of natural sign

18.5.3 Transaction Type Reporting Structure

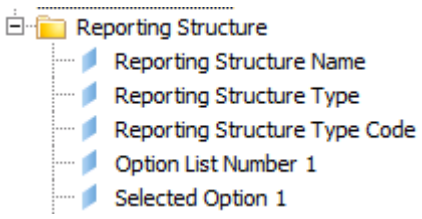
1. In the window **Option Lists**, locate the option list `Standard Reporting Structures`, we created above.
2. Add an option `BPM Transaction Types` to the list.
3. In the window **Reporting Structures**, create a new reporting structure on the dimension **Transaction Type**
4. Add a number of lines referencing existing transaction types and assign them to some groups.
5. Post a number of records (e.g. using the window **General Journal**) referencing transaction types listed in the reporting structure.

This reporting structure can be used both for reporting on the financial data we posted, in a grouped fashion, and for filtering to only include financial data on the transaction types we included in the reporting structure.

18.6 Business Layer

The business layer of the Reporting Structure consists solely of dimensional objects. There are objects for reporting on the name, type and other identity values of reporting structures. There are also objects for reporting on groups, sorting and signs defined on the individual lines of reporting structures.

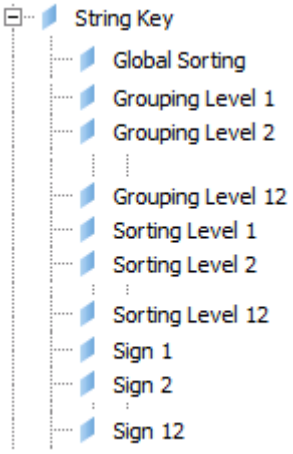
The [Reporting Structure] folder contains the central dimension objects for identifying reporting structures; whether by their name, type or option lists and option value.



[Reporting Structure Name]	REPORTINGSTRUCTUREHEADER.NAME
[Reporting Structure Type]	REPORTINGSTRUCTUREHEADER.REPORTINGSTRUCTURETYPE
[Reporting Structure Type Code]	EXREPORTINGSTRUCTUREHEADER.REPORTINGSTRUCTURETYPEPN
[Option List Number 1]	REPORTINGSTRUCTUREHEADER.OPTIONLISTNUMBER1
[Selected Option 1]	REPORTINGSTRUCTUREHEADER.SELECTEDOPTION1

18.6.1 Grouping, Sorting and Sign dimensions

The [Reporting Structure] folder also contains the dimensions for identifying, sorting, grouping and working with signs on any dimension. The [String Key] is the object that displays the dimension key value. It could be the customer number for a structure on customers, the account number for a structure on accounts, or something similar. The [Global Sorting] displays the overall sorting value across all groups.

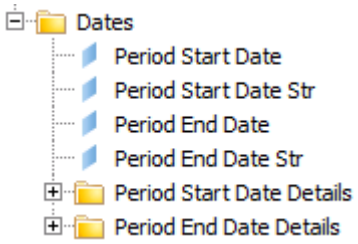


[String Key]	REPORTINGSTRUCTURELINE.GLOBALSORTING
[String Key]	REPORTINGSTRUCTURELINE.STRINGKEY
[Grouping Level 1]	REPORTINGSTRUCTURELINE.GROUPING1
[Grouping Level 2]	REPORTINGSTRUCTURELINE.GROUPING2

⋮
[Grouping Level 12] REPORTINGSTRUCTURELINE.GROUPING12
[Sorting Level 1] REPORTINGSTRUCTURELINE.SORTING1
[Sorting Level 2] REPORTINGSTRUCTURELINE.SORTING2
⋮
[Sorting Level 12] REPORTINGSTRUCTURELINE.SORTING12
[Sign Level 1] REPORTINGSTRUCTURELINE.SIGN1
[Sign Level 2] REPORTINGSTRUCTURELINE.SIGN2
⋮
[Sign Level 12] REPORTINGSTRUCTURELINE.SIGN12

18.6.2 Dates

Reporting structures can be defined to be valid for a specific date interval. The [Dates] folder contains dimension objects for reporting or restricting on these date interval by means of *from* and *to* date objects.



The objects are defined on from/to aliases of EXCALENDARDAYPV but the fields actually originate from the period start/end date fields on the reporting structure header to which the from and to aliases are joined.

[Period Start Date] EXREPORTINGSTRUCTUREHEADER.PERIODSTARTDATE
[Period Start Date Str] EXREPORTINGSTRUCTUREHEADER.PERIODSTARTDATEDS
[Period End Date] EXREPORTINGSTRUCTUREHEADER.PERIODENDDATE
[Period End Date Str] EXREPORTINGSTRUCTUREHEADER.PERIODENDDATEDS

18.7 Reporting Examples

In the following, we shall show how to use the Reporting Structure universe in practice for structuring and filtering data fetched through other BPM Reporting universes. However, before we start, let us outline the general methodology for combining data sets from the Reporting Structure and other universes.

18.7.1 Working with reporting structures

To understand how to work with reporting structure queries, we first need to understand how WebIntelligence works with queries to different universes. Each query in a WebIntelligence document, provides a data set. These data sets are not compatible until we tell WebIntelligence how to combine them. This is done by merging dimensions. Thus, a data set coming from a query to the Job Invoicing universe can be combined with a data set coming from a query to the Job Budgeting universe by merging dimensions from the two queries; e.g. [`Company No.`], [`Customer No.`] and [`Job No.`]

Typically, we want to use the merged dimension object in the data table. We will then get entries from both data sets despite whether a corresponding record exists from the other query. So technically, we get a kind of *outer join* between the two data sets.

When working with reporting structures, however, it is more likely that we want a kind of inner join. I.e., we wish only to see the reporting structure groups for which we have records in the other measure-bearing data set. So combining a data set coming from a query to the Reporting Structure with a data set coming from a query to the Finance universe merged on [`Account No.`] should not include groups for which no finance data exists.[2]

Furthermore, we typically want to section on the grouping objects—[`Grouping 1`], [`Grouping 2`], etc.—coming from the reporting structure query, and that is not possible in WebIntelligence. It is only possible to define sections on grouped dimensions when using data from multiple queries in the same block.

Best Practice Method: Combining Reporting Structure with other data (Outer)

Dimension objects which only exist in one of the queries and which is not mergeable, should be represented by dimension variables.

1. Create a query to the Reporting Structure universe.
2. Include the objects:

   [`String Key`]

   [`Grouping Level 1`]

[2] Later, we shall see an example where this is actually the desired result.

[Grouping Level 2]

[Grouping Level 3]

3. Create dimension variables for each grouping level object.
4. Create another query to some other universe.
5. Include at least one dimension from this universe. We shall call it the structure dimension.
6. Include measures objects from the universe as well.
7. Merge the structure dimension and [String Key] object.
8. Include a filter to select reporting structure (cf. Section 18.3)
9. For the report table, select:

   The merged dimension object

   [Grouping Level 1]

   [Grouping Level 2]

   [Grouping Level 3]

   The measure objects

This will display the union of values for the structure dimension and the [String Key]. Thus, this corresponds to an outer join.

Example 18.1 (Grouping finance data with the outer-join approach) *In this example, we shall apply the method using the Finance universe as data foundation for the measures. The structure dimension is [Account No.]*

The result of applying the method gives us a report that states the union of accounts from the two data sets; see Table 18.5. ■

Best Practice Method: Combining Reporting Structure with other data (Inner)

This method shows how to combine a data set from Reporting Structure with the data set of another universe, such that only dimensions for which there are measure data, are included.

1. Create a query to the Reporting Structure universe.
2. Include the objects:

   [String Key]

   [Grouping Level 1]

String Key	vnGrouping1	vnGrouping2	vnGrouping3	Debit	Credit
1110	Net Profit	EBITDA	Gross Profit	0.00	3,240,000.00
1120	Net Profit	EBITDA	Gross Profit		
1130	Net Profit	EBITDA	Gross Profit		
1140	Net Profit	EBITDA	Gross Profit		
1145	Net Profit	EBITDA	Gross Profit		
1150	Net Profit	EBITDA	Gross Profit		
1155	Net Profit	EBITDA	Gross Profit		
1210	Net Profit	EBITDA	Gross Profit		
1220	Net Profit	EBITDA	Gross Profit		
1230	Net Profit	EBITDA	Gross Profit		
1235	Net Profit	EBITDA	Gross Profit		
1240	Net Profit	EBITDA	Gross Profit		

Table 18.5: Finance data structured by an outer approach using a reporting structure.

[`Grouping Level 2`]

[`Grouping Level 3`]

3. Create dimension variables for each grouping level object.
4. Create another query to some other universe.
5. Include at least one dimension from this universe. We shall call it the structure dimension.
6. Include measures objects from the universe as well.
7. Merge the structure dimension and [`String Key`] object.
8. Include a filter to select reporting structure (cf. Section 18.3)
9. For the report table, select:

   The structure dimension object

   [`Grouping Level 1`]

   [`Grouping Level 2`]

   [`Grouping Level 3`]

   The measure objects

Account No.	Grouping 1	Grouping 2	Grouping 3	Debit	Credit
1110	Net Profit	EBITDA	Gross Profit	0.00	3,240,000.00
1510	Net Profit	EBITDA	Gross Profit	3,240,000.00	3,240,000.00
1710	Net Profit	EBITDA	Gross Profit	1,620,000.00	0.00
1910	Net Profit	EBITDA	Gross Profit	1,620,000.00	1,620,000.00
2110	Net Profit	EBITDA	Operating Expenses	800.00	0.00
2290	Net Profit	EBITDA	Operating Expenses	0.00	1,620,000.00
2510	Net Profit	Depreciation & Amortization		1,922.22	0.00
2560	Net Profit	Other Income		10,000.00	0.00
5120	Assets	Fixed Assets		1,922.22	3,844.44
5190	Assets	Current Assets		0.00	179,500.00
5210	Assets	Current Assets		4,050,000.00	0.00
5410	Assets	Current Assets		3,240,000.00	3,240,000.00
5510-DKK	Assets	Current Assets		0.00	1,000.00
6150	Equity	Stock	Foreign Exchanges	0.00	0.00
6310	Liabilities	Current Liabilities		0.00	810,000.00
6320	Liabilities	Current Liabilities		200.00	0.00
7110	Liabilities	Long Term Debt		0.00	1,000.00
7115	Liabilities	Long Term Debt		1,000.00	0.00

Table 18.6: Finance data structured by an inner approach using a reporting structure.

Example 18.2 (Grouping finance data with the inner-join method) *In this example, we shall apply the method using the* Finance *universe as data foundation for the measures. The structure dimension is [*`Account No.`*]*

The result of applying the method gives us a report that only states accounts for which there are posted figures (see Table 18.6.) ■

Best Practice Method: Sectioning on Reporting Structure groups

This method shows how to combine a data set from Reporting Structure with the data set of another universe, and sectioning on the groups of the reporting structure data set.

1. Create a query to the Reporting Structure universe.
2. Include the objects:

   [`String Key`]

[Grouping Level 1]

[Grouping Level 2]

[Grouping Level 3]

3. Create a query to some other universe.
4. Include the dimension object we wish to apply the reporting structure on, and some measures. Let us call the dimension the *structure dimension*. Additional dimensions can also be included.
5. Merge [String Key] and the structure dimension object from the other query.
6. Define detail variables for each grouping object.

   Qualification: select Detail

   Associated dimension: select the object [String Key]

   Formula: enter the grouping level object
7. For the data table include: The merged dimension, the grouping detail variables, other dimensions and measures from the other query
8. Drag the grouping detail variables up above the table to form sections
9. If we do not want to see empty tables for combinations of groupings where nothing has been posted, enable the table property Hide when empty.
10. If we do not want to see the groups for such empty tables either, define on the sections a conditional hide on when the table is empty.

Example 18.3 (Sectioning finance data on account groups) *In this example, we shall apply the method using the Finance universe as data foundation for the measures. The structure dimension is again [Account No.]*

The result of applying the method gives us a report that displays the grouping values from the reporting structure as headers for tables with data belonging to the groups (see Table 18.7.) ■

R Notice that the order of data changed in Table 18.7. The reason is that we got data sorted by the grouping labels. This can be changed by using the sorting objects ([Sorting Level 1]..[Sorting Level 12]) or the [Global Sorting] object.

Best Practice Method: Sectioning on Reporting Structure groups and other dimensions

This method is a variant of the previous method where we defined sections on the reporting structure groups. In this method, we shall add another section based on a dimension from the measure-bearing query.

Assets

Current Assets

Account No.	Debit	Credit
5190	0.00	179,500.00
5210	4,050,000.00	0.00
5410	3,240,000.00	3,240,000.00
5510-DKK	0.00	1,000.00

Fixed Assets

Account No.	Debit	Credit
5120	1,922.22	3,844.44

Table 18.7: Finance data sectioned by reporting structure groups.

Consider the report from Section 18.7.1. Apply the following modifications:

1. Modify the measure bearing query to include a new dimension object. Let us call this dimension the *section dimension*.
2. Include the *section dimension* in the table
3. Drag the section table above the table (above the existing groups) to form a new top-level section.
4. I some reports the structure complexity may require that the *section dimension* is also part of the table. Then add it to the table again and mark it as a hidden dimension.

Example 18.4 (Sectioning finance data on account groups) *In this example, we shall apply the method using the* Finance *universe as data foundation for the measures. The structure dimension is again [*`Account No.`*]. As section dimension, we choose the [*`Company No.`*] object from the [*`Finance`*] universe. See Table 18.8* ■

1

Assets

Current Assets

Account No.	Debit	Credit
5190	0.00	179,500.00
5210	4,050,000.00	0.00
5410	3,240,000.00	3,240,000.00
5510-DKK	0.00	1,000.00

Fixed Assets

Account No.	Debit	Credit
5120	1,922.22	3,844.44

Table 18.8: Finance data sectioned by company number and reporting structure groups.

Chapter 19

Asset Universe

The Asset universe provides objects for reporting on asset dimensions and asset changes; including the book value of assets. It is possible to distinguish the different types of adjustments done to assets like **Asset Acquisitions**, **Improvements**, **Write-ups** and **Write-downs**, **Depreciations**, and **Asset Sale**. It is also possible to report on the **Book Value** of assets. Concerning depreciations, there are several kinds. First of all there is a distinction between normal **Depreciations** and **Tax Depreciations**. There are dedicated book value objects for each of these kinds of depreciations, as it does not make sense to mix reporting on depreciations with tax depreciations when it comes to book value. There are also objects for reporting on depreciations related to **Job Cost** and **G/L**, respectively. Job cost depreciation is when the depreciation references a job and the posting of the depreciation is set to result in the creation of job entries. The G/L depreciations are all other depreciations. Again for both of these types, we have objects for reporting on the normal depreciation and on the tax depreciation.

The universe also includes objects for reporting on the bare asset change and all the individual types of changes. Thereby, it is possible to easily put together convenient report variables for reporting on exactly the categories of changes desired in reports. Furthermore, the universe contains dimension objects both for reporting on the dimensions associated the assets themselves, as well as the dimensions associated the asset adjustments.

19.1 Prerequisites

In order to fully benefit from the Asset universe, Maconomy should be set up such that:

- The system parameter **Use Calendar Day** is enabled and the calendar spans all necessary dates.

19.2 Understanding Asset Adjustments

Asset adjustments are done by performing operations on the assets in Maconomy. Some of the kind of adjustments are *acquisition*, *write-ups* and *write-downs*, and *depreciations*. Each adjustment may change the value of the asset; the book value. Thus, acquiring an asset with a value of 5.000, 00, makes it have a book value of that amount. Depreciating it with 1.000, 00 will subtract that from the book value.

19.2.1 Asset Entries

When assets are adjusted, it results in the creation of entries in the database table `ASSETENTRY`. Central fields on that table are:

`THEAMOUNT` which states the amount with which we adjust the asset. The above mentioned depreciation would result in an asset entry with `THEAMOUNT` 1.000, 00.

`ASSETTRANSACTIONTYPE` which states the kind of adjustment that was performed. However, often several entries are created from one adjustment because these entries represent the double-entry bookkeeping that is going on for assets. Thus selling an asset will both create an entry of type `Sale` but also an entry of type `Acquisition` having the negative amount of the original acquisition value.

`TRANSACTIONSUBGROUP` which states in which adjustment context the entry was created. The entries that just state that we acquire, sell or depreciate an asset, will have a blank value in this field. However, the additional entries that are created to keep the double-entry bookkeeping working will state informative values in this field. E.g. the entry of type `Acquisition` having the negative value, will state `Sales` as sub-group. Thus, the two fields `ASSETTRANSACTIONTYPE` and `TRANSACTIONSUBGROUP` are both very important for categorizing the entries and interpreting what they mean when we report.

19.2.2 Book Value

The book value of an asset, is basically the summation of all the adjustments done to the asset from the beginning and up to a certain date. Some asset entries state the value as positive even though the adjustment decreases the book value. That is for instance the case with depreciations. The formula for calculating book value is thus:

$$\text{Book Value} = \begin{cases} \text{Acquisitions} + \text{Improvements} + \text{Write-Ups} \\ -\text{Write-Downs} - \text{Depreciations} - \text{Corrections} \end{cases}$$

Typcially, we want to calculate the book value as of a certain date. This should be done as follows:

$$\text{Current Book Value} = \begin{cases} \texttt{Acquisitions} + \text{Improvements} + \text{Write-Ups} \\ -\text{Write-Downs} - \text{Depreciations} - \text{Corrections} \\ \textit{where} \quad \text{Entry Date} \leq \text{Current Date} \end{cases}$$

Using a range restriction on *Entry Date* gives the change in book value within the selected date range.

R There are two sorts of deprecations in Maconomy and they should usually not be mixed when reporting on assets. Either one uses depreciations or otherwise tax depreciations. Therefore, the objects for reporting on book value come in two versions; one assuming depreciation and one assuming tax depreciation.

19.3 Maconomy Workflows

In the following sub-sections, we shall look at the common Maconomy workflows that are concerned with the data for which the Asset universe is made for reporting on.

19.3.1 Asset Acquisition and Disposal

In Maconomy, we talk about **Acquisitions** when assets are bought externally or internally. We talk about **Disposals** when assets are sold or thrown out. A workflow for acquiring and then disposing an asset through an asset sale could be as follows:

1. In the window **Assets**, create a new asset of type `Automobile`.
2. State a cost price (acquisition price) of $\$50.000,00$.
3. Select an acquisition date.
4. In the Depreciation Period field enter 60 and hit enter.

This will create an entry in the table `ASSET` and an entry in the table `ASSETENTRY`. The two entries are bound by the same `ASSETNUMBER`. The asset entry will have the `ASSETENTRY.ASSETTRANSACTIONTYPE Acquisition` (popup number 0). The `ASSETENTRY.TRANSACTIONSUBGROUP` will be blank. This means that we have only acquired the asset; we have not done anything additional with it.

1. In the window **Asset Disposal**, enter a date in the field `Date of Sales`.
2. Enter a sales amount of $\$40.000,00$ and state 100.0 in the field `Pct. Sold`. This means that we sold the asset for $\$40.000,00$ and that it is the whole automobile that was sold.
3. Enter a remark.
4. Hit Enter and apply the action `Approve Sale`.

Examine the ASSETENTRY table again. We see that we now got two additional asset entries; both having the value Sale in the field TRANSACTIONSUBGROUP:

- The first entry has ASSETTRANSACTIONTYPE Acquisition and a value of -$50.000,00. This means that we have reduced the value of our acquisition by the full amount that we originally acquired the asset for. Thus, if we summarize the amounts of the entries having ASSETTRANSACTIONTYPE Acquisition, we will get a 0 because we have no acquisition anymore. If we wish to know how much we acquired (disregarding whether it was sold again), we need to filter on that ASSETTRANSACTIONTYPE states Acquisition, that TRANSACTIONSUBGROUP is not Transfer or Relocation, and that the amount is positive (see the definition of the object [Acquisition Added]. If the amount was negative, it would mean a sale. If the TRANSACTIONSUBGROUP was Transfer or Relocation, it would mean that we acquired the asset through a transfer or relocation.
- The second entry has ASSETTRANSACTIONTYPE Sale and a value of $40.000,00. This means that we have sold the asset for a value of $40.000,00. If we wish to know how much has been sold (regardlessly of what has been acquired), we can filter to exactly this entry. The entry also states what the gain or loss of selling was. Prior to the sale, our asset had a book value of $50.000,00 because that was what we acquired it for and no adjustments had been made. However, selling the asset for $40.000,00 results in a loss of $10.000,00, which is also what is stated in the field LOSSGAINAMOUNT.

19.3.2 Asset Depreciation and Corrections

Typically, assets decrease in value over time. The speed and principles with which they do, is defined by the depreciation principle and number of periods assigned to the asset. If we choose the method Straight Line, by Month and 60 periods, it means that we reduce the book value of the asset by the same amount every month over 60 months; so after 5 years, the asset is fully depreciated (the book value is zero). Consider the following:

1. In the window **Assets**, create a new asset of type Automobile.
2. State a cost price (acquisition price) of $100.000,00.
3. Select an acquisition date which is 3 months back.
4. In the Depreciation Period field enter 60 and hit enter.

   This adds an asset with a book value of $100.000,00.
5. In the window **Depreciation Adjustment**, restrict to the asset in question and select an entry date two months back. This corresponds to apply one month depreciation.
6. Hit Enter and run the action OK.

This will create two asset entries. The first one we already know as the one representing the fact that we acquired the asset. The second entry has `ASSETTRANSACTIONTYPE Depreciation`. It's value is $\frac{1}{60}$ of the original book value of $\$100.000,00$. To calculate the new book value, we thus need to summarize the value of the two asset entries created so far.

19.3.3 Asset Transfer

Assets can be transferred from one company to another company within the same enterprise. In Maconomy, this is called an **Asset Transfer**. A workflow involiving an asset transfer could look like this:

1. In the window **Assets**, create a new asset of type `Automobile`.
2. State the asset number `1001`.
3. State a cost price (acquisition price) of $\$60.000,00$.
4. Select an acquisition date.
5. In the `Depreciation Period` field, enter 60 and hit enter.

   As we have seen before, this creates an asset entry of type `Acquisition` with a blank sub group.
6. In the window **Asset Disposal**, state a new asset number `1002` in the field `New Asset No.`
7. Select a disposal date for the transfer.
8. Select an asset group. The same group `Automobiles` will be fine here.
9. Enter a description for the transfer.
10. Run the action `Approve Transfer`.

When examining the asset entries with the old asset number `1001`, we see that we got two additional entries. The first one has type `Acquisition` and is the offset entry to the acquisition of the asset. It thus has the negative amount -$\$60.000,00$. It means that we removed our asset from our collection of assets. The second entry has the type `Transfer` (same for the sub group) but no amount. This record is not relevant from a reporting point of view. It just states that we transferred the asset, but we already know that from the other entry. Summarizing the amounts of the entries for asset `1001` now states a book value of 0 because we transferred all of the asset to a new asset number.

Examining the asset entries with the new asset number `1002`, we see just one entry. This entry has the type `Acquisition` and sub group `Transfer`. The amount is positive (contrary as the similar entry for asset number `1001`), and this means that we basically acquired this as a new asset, but by means of a transfer.

If several adjustments have been made to asset number `1001` prior to the transfer, we would have seen more offset entries. So if depreciations have been applied, then when transferring, we would get offset entries of type `Depreciation` and sub group `Transfer`.

We can summarize the basic principles of asset adjustments as follows:

- Asset values are either *Added*, *Transferred* or *Disposed*. E.g. a disposed improvement is the value of an improvement that we once made, but that we now loose because are getting rid of the asset.
- The field `ASSETTRANSACTIONTYPE` states this type of value change.
- For some types the field `TRANSACTIONSUBGROUP` is used for stating whether it is in context of a transfer or relocation.
- Adjustment amount states the *direction* of the value; e.g. bought/sold or transferred to/from.
- When assets are disposed, a trace of adjustments are added as entries (a kind of double-bookkeeping). E.g. when selling an asset, we get offset entries for all that has been done to the asset through time: Acquiring it, improving it, depreciating it, etc. This ensures that after the sale, the book value of the asset is zero.

19.4 Business Layer

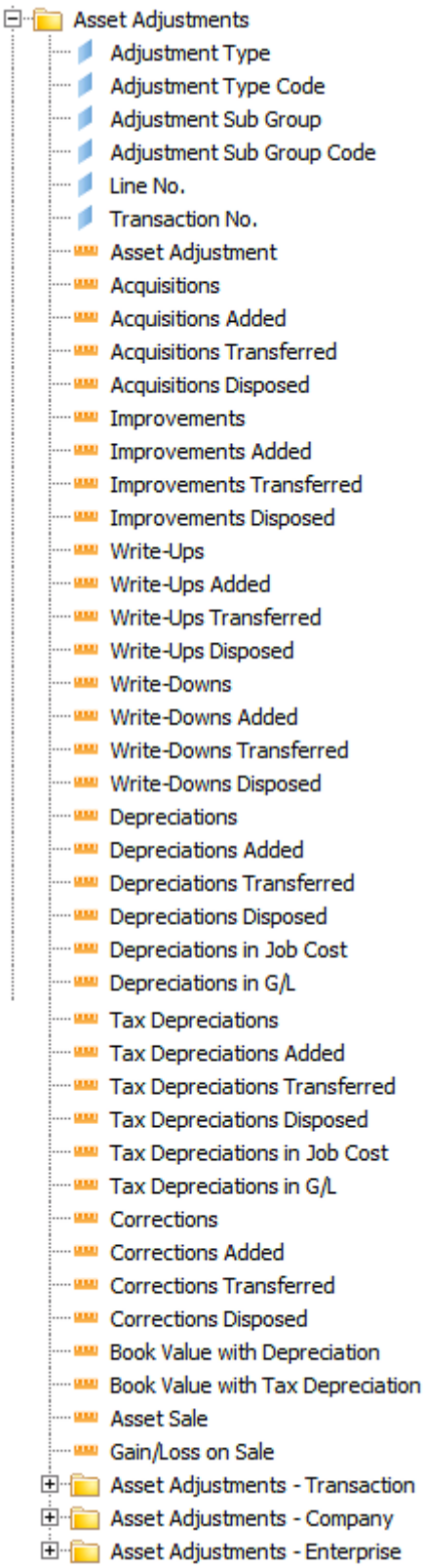
The business layer is centered around measures for reporting on each of the different types of change that can be done to an asset like acquiring it, disposing it (i.e. selling or throwing it out), depreciating it, transferring it, or doing other operations that can change the book value of the asset. There are also dedicated measures for reporting on the current book value of the assets. Furthermore, the business layer includes dimensions that are central when reporting on assets, like company, standard dimensions, different kinds of dates, and information about the assets and asset groups. In addition, there are dimensions for reporting on the type of asset adjustment, the transaction type of the adjustment and the sub group of the adjustment. These can be combined with the measure object displaying the asset change. Thereby, it is possible to compose any desired way of reporting or filtering on the asset adjustments done in Maconomy. This is convenient if the many predefined measures—after all—do not fit the specific needs.

In the following, we shall explain in more detail, the different measures of the business layer.

19.4.1 Asset Adjustments

The dimension and measure objects for reporting on asset adjustments, are located in the [`Asset Adjustments`] folder.

For each type of adjustment (acquisition, sale, depreciation, etc.), there are four different measure objects to use in report. The first one simply displays the value of the given type of adjustment. E.g. [`Acquisitions`] displays the value of all adjustments related to acquisitions (both adding assets, disposing them, and transferring them). The second one displays the value of added adjustments only. The third one displays the transferred value related to the adjustment type. The fourth one displays the value disposed. The folder also contains the measure object [`Adjustment Value`] which displays the adjustment value and is to be coupled with relevant dimensions like [`Adjustment Type`] and [`Adjustment Sub Group`] to achieve the desired filtering on asset adjustments.
Furthermore, the folder contains two sorts of measures for book value; one assuming the use of depreciation and another assuming tax depreciation. When reporting on book value, the book value measure object must be combined with a restriction filtering on dates (for more information see the reporting examples). The folder also contains the object [`Gain/Loss on Sale`]. This object shows the gain or loss in case the asset adjustment is a sale.
All the mentioned measure objects are available in the currency of the transaction, in company currency and in the currency of the enterprise. Furthermore, the objects are available in forms where the objects prompt the user for selecting a currency type (*Transaction*, *Company* or *Enterprise*).



[Asset Adjustment - Company] ASSETENTRY.THEAMOUNT
[Acquisitions] ASSETENTRY.THEAMOUNT *if* ASSETENTRY.ASSETTRANSACTIONTYPEPN $\in (0, 9)$ 0 *otherwise*
[Acquisitions Added] ASSETENTRY.THEAMOUNT *if* ASSETENTRY.ASSETTRANSACTIONTYPEPN $\in (0, 9)$ *and* ASSETENTRY.TRANSACTIONSUBGROUPPN $\notin (7, 8)$ *and* ASSETENTRY.THEAMOUNT $>= 0$ 0 *otherwise*
[Acquisitions Transferred] ASSETENTRY.THEAMOUNT *if* ASSETENTRY.ASSETTRANSACTIONTYPEPN $\in (0, 9)$ *and* ASSETENTRY.TRANSACTIONSUBGROUPPN $\in (7, 8)$ 0 *otherwise*
[Acquisitions Disposed] ASSETENTRY.THEAMOUNT *if* ASSETENTRY.ASSETTRANSACTIONTYPEPN $\in (0, 9)$ *and* ASSETENTRY.TRANSACTIONSUBGROUPPN $\notin (7, 8)$ *and* ASSETENTRY.THEAMOUNT < 0 0 *otherwise*
[Improvements] ASSETENTRY.THEAMOUNT *if* ASSETENTRY.ASSETTRANSACTIONTYPEPN $= 1$ 0 *otherwise*
[Improvements Added] ASSETENTRY.THEAMOUNT *if* ASSETENTRY.ASSETTRANSACTIONTYPEPN $= 1$ *and* ASSETENTRY.TRANSACTIONSUBGROUPPN $\notin (7, 8)$ *and* ASSETENTRY.THEAMOUNT $>= 0$ 0 *otherwise*
[Improvements Transferred] ASSETENTRY.THEAMOUNT *if* ASSETENTRY.ASSETTRANSACTIONTYPEPN $= 1$ *and* ASSETENTRY.TRANSACTIONSUBGROUPPN $\in (7, 8)$ 0 *otherwise*

[Improvements Disposed]
ASSETENTRY.THEAMOUNT
if ASSETENTRY.ASSETTRANSACTIONTYPEPN $= 1$
and ASSETENTRY.TRANSACTIONSUBGROUPPN $\notin (7, 8)$
and ASSETENTRY.THEAMOUNT < 0
0 *otherwise*

[Write-Ups]
ASSETENTRY.THEAMOUNT
if ASSETENTRY.ASSETTRANSACTIONTYPEPN $= 3$
0 *otherwise*

[Write-Ups Added]
ASSETENTRY.THEAMOUNT
if ASSETENTRY.ASSETTRANSACTIONTYPEPN $= 3$
and ASSETENTRY.TRANSACTIONSUBGROUPPN $\notin (7, 8)$
and ASSETENTRY.THEAMOUNT $>= 0$
0 *otherwise*

[Write-Ups Transferred]
ASSETENTRY.THEAMOUNT
if ASSETENTRY.ASSETTRANSACTIONTYPEPN $= 3$
and ASSETENTRY.TRANSACTIONSUBGROUPPN $\in (7, 8)$
0 *otherwise*

[Write-Ups Disposed]
ASSETENTRY.THEAMOUNT
if ASSETENTRY.ASSETTRANSACTIONTYPEPN $= 3$
and ASSETENTRY.TRANSACTIONSUBGROUPPN $\notin (7, 8)$
and ASSETENTRY.THEAMOUNT < 0
0 *otherwise*

[Write-Downs]
ASSETENTRY.THEAMOUNT
if ASSETENTRY.ASSETTRANSACTIONTYPEPN $= 4$
0 *otherwise*

[Write-Downs Added]
ASSETENTRY.THEAMOUNT
if ASSETENTRY.ASSETTRANSACTIONTYPEPN $= 4$
and ASSETENTRY.TRANSACTIONSUBGROUPPN $\notin (7, 8)$
and ASSETENTRY.THEAMOUNT $>= 0$
0 *otherwise*

[Write-Downs Transferred]
ASSETENTRY.THEAMOUNT
if ASSETENTRY.ASSETTRANSACTIONTYPEPN $= 4$
and ASSETENTRY.TRANSACTIONSUBGROUPPN $\in (7, 8)$
0 *otherwise*

[Write-Downs Disposed] ASSETENTRY.THEAMOUNT *if* ASSETENTRY.ASSETTRANSACTIONTYPEPN $= 4$ *and* ASSETENTRY.TRANSACTIONSUBGROUPPN $\notin (7, 8)$ *and* ASSETENTRY.THEAMOUNT < 0 0 *otherwise*
[Depreciations] ASSETENTRY.THEAMOUNT *if* ASSETENTRY.ASSETTRANSACTIONTYPEPN $= 2$ 0 *otherwise*
[Depreciations Added] ASSETENTRY.THEAMOUNT *if* ASSETENTRY.ASSETTRANSACTIONTYPEPN $= 2$ *and* ASSETENTRY.TRANSACTIONSUBGROUPPN $\notin (7, 8)$ *and* ASSETENTRY.THEAMOUNT $>= 0$ 0 *otherwise*
[Depreciations Transferred] ASSETENTRY.THEAMOUNT *if* ASSETENTRY.ASSETTRANSACTIONTYPEPN $= 2$ *and* ASSETENTRY.TRANSACTIONSUBGROUPPN $\in (7, 8)$ 0 *otherwise*
[Depreciations Disposed] ASSETENTRY.THEAMOUNT *if* ASSETENTRY.ASSETTRANSACTIONTYPEPN $= 2$ *and* ASSETENTRY.TRANSACTIONSUBGROUPPN $\notin (7, 8)$ *and* ASSETENTRY.THEAMOUNT < 0 0 *otherwise*
[Depreciations in Job Cost] ASSETENTRY.THEAMOUNT *if* ASSETENTRY.ASSETTRANSACTIONTYPEPN $= 2$ *and* ASSETENTRY.POSTDEPRECIATIONASJOBENTRYPN $= 1$ *and* ASSETENTRY.JOBNUMBER <> ‘ ‘ 0 *otherwise*
[Depreciations in G/L] ASSETENTRY.THEAMOUNT *if* ASSETENTRY.ASSETTRANSACTIONTYPEPN $= 2$ (*and* ASSETENTRY.POSTDEPRECIATIONASJOBENTRYPN $= 0$ *and* ASSETENTRY.JOBNUMBER $=$ ‘ ‘) 0 *otherwise*

[Tax Depreciations]
ASSETENTRY.THEAMOUNT
if ASSETENTRY.ASSETTRANSACTIONTYPEPN = 10
0 *otherwise*

[Tax Depreciations Added]
ASSETENTRY.THEAMOUNT
if ASSETENTRY.ASSETTRANSACTIONTYPEPN = 10
and ASSETENTRY.TRANSACTIONSUBGROUPPN $\notin (7, 8)$
and ASSETENTRY.THEAMOUNT >= 0
0 *otherwise*

[Tax Depreciations Transferred]
ASSETENTRY.THEAMOUNT
if ASSETENTRY.ASSETTRANSACTIONTYPEPN = 10
and ASSETENTRY.TRANSACTIONSUBGROUPPN $\in (7, 8)$
0 *otherwise*

[Tax Depreciations Disposed]
ASSETENTRY.THEAMOUNT
if ASSETENTRY.ASSETTRANSACTIONTYPEPN = 10
and ASSETENTRY.TRANSACTIONSUBGROUPPN $\notin (7, 8)$
and ASSETENTRY.THEAMOUNT < 0
0 *otherwise*

[Tax Depreciations in Job Cost]
ASSETENTRY.THEAMOUNT
if ASSETENTRY.ASSETTRANSACTIONTYPEPN = 10
and ASSETENTRY.POSTDEPRECIATIONASJOBENTRYPN = 1
and ASSETENTRY.JOBNUMBER <> ‘ ‘
0 *otherwise*

[Tax Depreciations in G/L]
ASSETENTRY.THEAMOUNT
if ASSETENTRY.ASSETTRANSACTIONTYPEPN = 10
(*and* ASSETENTRY.POSTDEPRECIATIONASJOBENTRYPN = 0
and ASSETENTRY.JOBNUMBER = ‘ ‘)
0 *otherwise*

[Corrections]
ASSETENTRY.THEAMOUNT
if ASSETENTRY.ASSETTRANSACTIONTYPEPN = 5
0 *otherwise*

[Corrections Added]	ASSETENTRY.THEAMOUNT *if* ASSETENTRY.ASSETTRANSACTIONTYPEPN $= 5$ *and* ASSETENTRY.TRANSACTIONSUBGROUPPN $\notin (7, 8)$ *and* ASSETENTRY.THEAMOUNT $>= 0$ 0 *otherwise*
[Corrections Transferred]	ASSETENTRY.THEAMOUNT *if* ASSETENTRY.ASSETTRANSACTIONTYPEPN $= 5$ *and* ASSETENTRY.TRANSACTIONSUBGROUPPN $\in (7, 8)$ 0 *otherwise*
[Corrections Disposed]	ASSETENTRY.THEAMOUNT *if* ASSETENTRY.ASSETTRANSACTIONTYPEPN $= 5$ *and* ASSETENTRY.TRANSACTIONSUBGROUPPN $\notin (7, 8)$ *and* ASSETENTRY.THEAMOUNT < 0 0 *otherwise*

Notice that not all possible values of the popup ASSETTRANSACTIONTYPEPN are used. We do not distinguish on Sale, Transfer nor Relocation. Sale—having popup number 6—is indirectly handled by acquisitions being disposed. These we find by the fact that their value is negative. The entry of type Sale is thus only needed when we need the specific asset entry stating the value we sold the asset for and thus for stating the gain and loss of the asset.

Transfer and Relocation—having popup number 7 and 8—are used only on the sub group. Thus, if we wish to report on the transfer of an asset, we should add all the different transfer objects. Similarly as if we want to report on all the additions; then we should add all the addition objects.

19.5 Reporting Examples

19.5.1 Asset acquisitions and disposals

Example 19.1 (Reporting on asset acquisitions and disposals) *In this example, we shall build a small report that displays assets and their additions and disposals within a date period.*

1. *Create a new WebIntelligence document and select eh universe* Asset
2. *Add the following objects for the query:*

   [**Asset No.**] *(folder [Asset Information])*

   [**Asset Descr.**] *(folder [Asset Information])*

[Currency] *(folder [Currency])*

[Acquisitions Added] *(folder [Asset Adjustments])*

[Acquisitions Disposed] *(folder [Asset Adjustments])*

[Asset Sale] *(folder [Asset Adjustments])*

[Gain/Loss On Sale] *(folder [Asset Adjustments])*

3. *Add the [Entry Date]] (folder [Dates]) object as a filter using the between operation and prompting the user for a from and a to date.*
4. *Use the object [Currency] to define a section.*
5. *Refresh the report.*

When running the report, it will prompt for selecting a currency type and a from and a to date. The report will display a list of assets and for each show the acquisition values, the value disposed, the price they were sold for and the gain/loss in context of the sale. Only the assets that were acquired or disposed within the date range, will be included. ■

19.5.2 Asset depreciation

For companies that both work with depreciation and tax depreciation, we may want to define the report to handle both. This can be done by including both kind of measures as well as using the dimension object [Use Tax Depreciation]. This object will prompt the user for whether to use tax depreciation and a choice can then be made between the two kinds of measures in the report.

Example 19.2 (Reporting on asset depreciations) *In this example, we shall build a small report that that shows depreciation of assets with the distinction between normal depreciation and tax depreciation. We shall also show how to break down the depreciation value into job cost and G/L related.*

1. *Create a new WebIntelligence document and select eh universe Asset*
2. *Add the following objects for the query:*

[Asset No.] *(folder [Asset Information])*

[Asset Descr.] *(folder [Asset Information])*

[Currency] *(folder [Currency])*

[Depreciation Added] *(folder [Asset Adjustments])*

[Depreciation in Job Cost] *(folder [Asset Adjustments])*

[Depreciation in G/L] *(folder [Asset Adjustments])*

[Tax Depreciation Added] *(folder [Asset Adjustments])*

[Tax Depreciation in Job Cost] *(folder [Asset Adjustments])*

[Tax Depreciation in G/L] *(folder [Asset Adjustments])*

[Use Tax Depreciation] *(folder [Report])*

3. *Add the [Entry Date]] (folder [Dates]) object as a filter using the between operation and prompting the user for a from and a to date.*
4. *Use the object [Currency] to define a section.*
5. *Create a variable vnDepreciation which cases on the user value of the object [User Tax Depreciation]. If the user answers Yes, the [Depreciation Added] object should be used; else the [Tax Depreciation Added] object should be used.*
6. *Create two additional variable with the same approach for displaying depreciation in Job Cost and in G/L.*
7. *Add the variables as columns to the report and remove the existing measure objects. Also remove the column showing the object [Use Tax Depreciation].*
8. *Refresh the report.*

The report will prompt the user for selecting a currency type and for whether to display depreciation or tax depreciation. ■

Example 19.3 (Including corrections to depreciations) *Corrections to depreciations can also be performed in Maconomy. They can be reported on by including the [Corrections Added] object. Consider the following modification to the report from Example 19.2.*

1. *Add the object [Corrections Added]] (folder [Asset Adjustments]) to the query.*
2. *In the variable vnDepreciation add the correction object to the depreciation object.*

■

19.5.3 Asset Book Value

When reporting on the book value, we must decide a date interval for summation. The book value objects just summarizes all of the adjustments made to an asset, and it is thus our job to apply the desired restriction. If we wish to report on the book value as of a certain date, we should prompt for that date and apply it as a restriction. If we wish to report on the change in book value within a date range, we should prompt for a from and to date, and apply a between operation as filter.

Example 19.4 (Reporting on Asset Book Value) *In this example, we shall create a report which displays the book value of assets as of a given date.*

1. *Create a new WebIntelligence document and select eh universe* Asset
2. *Add the following objects for the query:*

   [Asset No.] *(folder [Asset Information])*

   [Asset Descr.] *(folder [Asset Information])*

   [Currency] *(folder [Currency])*

   [Book Value with Depreciation] *(folder [Asset Adjustments])*
3. *Use the object [Currency] to define a section.*
4. *Add the [Entry Date]] (folder [Dates]) object as a filter prompting the user for a date and applying* less than or equal *to that date.*
5. *Refresh the report.*

The report will prompt the user for a date and then display a list of assets and their book value as of the selected date. ■

Example 19.5 (Reporting on Asset Book Value quarterly) *Another way to get an overview of the assets, is to show the book value by quarter for a year.*

1. *Create a new WebIntelligence document and select eh universe* Asset
2. *Add the following objects for the query:*

   [Asset No.] *(folder [Asset Information])*

   [Asset Descr.] *(folder [Asset Information])*

   [Currency] *(folder [Currency])*

   [Book Value with Depreciation] *(folder [Asset Adjustments])*

   [Quarter Short] *(folder [Entry Date Details])*
3. *Use the object [Currency] to define a section.*
4. *Add the [Year]] (folder [Entry Date Details]) object as a filter prompting the user for a year and applying* equal *to that year.*
5. *Change the default horizontal table to a cross-tab table. Use the dimension object [Quarter Short] horizontally, the objects [Asset No.] and [Asset Descr.] vertically, and the object [Book Value with Depreciation] in the table cells.*
6. *Refresh the report.*

The report will show for a given year, how the book value of assets may change by quarter. More complicated displays or filtering can also be used. ■

Chapter 20

Opportunity Universe

The Opportunity universe provides objects for reporting on opportunities—i.e. sales pipeline—in Maconomy. It is possible to report on the individual opportunities and the contact companies associated. It is also possible to distribute the opportunities into quarters, half-years or other timeline categories in order to get an overview of potential, future revenue. It is possible to categorize the opportunities into the event stages and report on the event flow associated opportunities in general. It is also possible to report on the probability with which the sales prices are weighted in Maconomy. Combined with queries to the Job Budgeting universe, it is possible to report on budgets made on opportunities.

The measure objects can be grouped and restricted by a variety of dimensions including contact company, contact person, job, employee, dates, and the ten standard dimensions.

20.1 Prerequisites

In order to fully benefit from the Opportunity universe, Maconomy should be set up such that:

- The system parameter **Use Calendar Day** is enabled and the calendar spans all necessary dates.

Chapter 21

Subscription Universe

The Subscription universe provides objects for reporting on subscribers and subscription orders. The universe is centered on subscription order information including individual subscription orders, corresponding price, dates, and the quantities. Combined with queries to the Sales Order universe, it is possible to report on the full workflow of subscriptions from the definition and order, to the invoices. The universe can also be used in combination with queries to the Job Invoicing universe if invoicing of subscriptions is done in the Job Cost module.

The measure objects can be grouped and restricted by a variety of dimensions including company, customer, dates, and the ten standard dimensions.

21.1 Prerequisites

In order to fully benefit from the Subscription universe, Maconomy should be set up such that:

- The system parameter **Use Calendar Day** is enabled and the calendar spans all necessary dates.

Chapter 22

Sales Order Universe

The Sales Order universe provides objects for reporting on sales orders and corresponding invoices performed in the sales order module of Maconomy. It is possible to report on orders being in the different stages; from quotes via sales orders or back orders to delivery and invoicing. Hence, it is possible to compare prices and quantities through the different stages of a sales order flow. It is possible both to report on the total quantities and prices, and the contribution from the individual order lines. Here, the totals rarely make up the sum of the lines as charges and invoice discounts may be different on line and total level. Combined with queries to the Subscription universe, it is possible to report on the full flow of subscription orders if these are invoiced in the sales order module.

The measure objects can be grouped and restricted by a variety of dimensions including company, customer, order type, dates, and the ten standard dimensions.

22.1 Prerequisites

In order to fully benefit from the Sales Order universe, Maconomy should be set up such that:

- The system parameter **Use Calendar Day** is enabled and the calendar spans all necessary dates.

Chapter 23

Event Universe

The Event universe provides objects for reporting on events, event flows and data associated events. The universe does not provide measure objects but is entirely for reporting on dimensions. It can be combined summarized amounts as measures. The universe is for reporting on dimensional data only, but can be used in combination with queries to other universes to provide more detailed information about events.

23.1 Prerequisites

In order to fully benefit from the Event universe, Maconomy should be set up such that:

- The system parameter **Use Calendar Day** is enabled and the calendar spans all necessary dates.

Chapter 24

System Universe

The System universe provides objects for reporting on central Maconomy setup, installation and upgrade information. It is possible report on the system maintenance log to get insight into the version and service pack level, as well as th status on or during upgrading of Maconomy. It is also possible to report on central system information parameters like year-end result account and the dimension to be used for local charts of accounts. Furthermore, it is possible to report out of the **Query Log** and **Upgrade Log**, which can be convenient when investigating the Maconomy system for transactions. In addition, it is possible to report on the current version and service pack, as well as changes to time sheet and expense sheets after submission.

The System universe is a universe for administrators and **not** for any end-user.

24.1 Prerequisites

The use of the universe requires no prerequisites.

Chapter 25

User Information Universe

The User Information universe provides objects for report on user and employee setup.

It is possible to report on various properties of user setup like administrator rights and action capabilities. It is also possible to report on various properties of the user like what the associated employee is, who the supervisor, tutor and secretary is.

Furthermore, it is also possible to report on who the current user and employee is who is running the report in question. Thereby, the universe is convenient to use in any reporting situation where reporting data or restrictions need to be tied to the current user or employees associated the current user.

25.1 Prerequisites

The use of the universe requires no prerequisites.

Part III

BPM Analysis

Chapter 26

General Design Principles

This chapter explains the general design principles of the BPM Analysis universes. The succeeding sections will describe each universe. For each universe, first give an overall description of what the universe contains and can be used for. Then we describe some typical Maconomy workflows and explain how the data can be reported on using the universe.

26.1 Joins

The business logic concerning joins described in Section 3.1 also applies to the BPM Analysis universes. That is, most of the joins that are used in between tables have a cardinality of *many-to-one*. Typically for most dimensions directly associated a fact there is an inner join because the fact record always has a value for that dimension, and typically for dimensions of dimensions this is not the case. In BPM Reporting, when we do not necessarily have a dimension value, the join becomes a left-outer join instead of an inner join. This is different in BPM Analysis; here all joins are inner joins. The way this is done is by making sure that all dimensions include a blank record (*null-row*) joins. The facts which have a dimension field which can be blank, inner joins to the dimension table, and that join will associate the blank row with the fact record. Similar for dimensions on dimensions.

26.2 Historical Dimensions

The data warehouse on which the BPM Analysis universes are defined, is able to store changes to dimensions. Loading the data warehouse with Maconomy data is performed by the *so-called ETL*(*Extract-Transform-Load*) jobs of the data warehouse. The jobs read data (*Extract*) in Maconomy, transforms it (*Transform*) and stores it (*Load*) in the database tables of the data warehouse. Here the data is stored so they are optimal for reporting.

When reading in dimensions it may be that a field has changed since the last load. E.g. a vendors name may have changed even though it is the same vendor with the same vendor number. The data warehouse keeps track of that change such that it is possible to report on e.g. vendor invoices issued when the vendor had the old name and when the vendor has the new name. This is valuable information for many kinds analysis. Dimensions which change like this and are tracked in the data warehouse, are called *Slowly Changing Dimensions* or simply *Historical Dimensions*. This feature is one of the special features of BPM Analysis compared to BPM Reporting. However, many dimensions Maconomy keeps track of already: employee positions, budget revisions, etc. For such dimensions, history is applicable in both BPM products.

26.3 Dates

In BPM Analysis, all data are stored in the data warehouse so that they are optimal for reporting. This also concerns dates which are stored as genuine database date types. In Maconomy, dates are represented as strings. This requires that in several cases we need in BPM Reporting to convert these strings to genuine dates in order for BusinessObjects tools to process them. E.g. if the user wishes to restrict on a date when running a report, the date we select in WebIntelligence is of a genuine database date type. In some cases, records have date fields without there being a date. Replacing with a null date is not always a good idea because the date may be used for looking up the right historical dimensions. In such cases, we change that date to be either the systems current date or the maximum date.

26.4 Popups

For popups, we also utilize that we can store data in the data warehouse so they are optimal for performance. When reading data from Maconomy, we typically do that through the EX view which provides the external data values for popups in addition to the enumerated values which are integers. Typically, we then store the external data values in the data warehouse. Also, see more explanation of this in Section 3.2.

The external data values stored in the data warehouse are not the values stated in the corresponding popup values in Maconomy. The reason is that the popup values in Maconomy may be changed to fit the enterprise language of the customer's system, and then standard reports (assuming English terms) would not work. Therefore all terms (typically `"Yes"` and `"No"`) are kept in English. In reports, they can always be redefined for display by means of variables.

26.5 Access Control

In BPM Analysis, access control is only applied on fact tables. When reporting on a BPM Analysis universe, most likely some fact is involved and the universe will make sure that

the proper access control view is applied to filter the data that the user is allowed to see. The access control principles are the same as they are for the corresponding Maconomy tables in Maconomy and in BPM Reporting.

26.6 Other Aspects

The other aspects like objects prompting for currency, hierarchies, naming conventions, etc. that was explained in the design principles of BPM Reporting (Section 3.5 to Section 3.8). The differences to BPM Analysis are only minor; e.g., company currency is referred to as *base* and job currency is referred to as *currency* which is closer to the naming applied on the database level.

Chapter 27

Job Cost Universe

The Job Cost universe provides objects for reporting on job cost registrations, job invoices, job invoices on account, job budgets, forecast budgets, employee utilization, and purchase orders in relation to job budgets. It is possible to report on quantity, cost prices and billing prices in relation to registrations and invoices. It is possible to report on the up/down writings, the revenue recognized, accruals and the invoiced or reconciled amounts (thereby possible to deduce the Net on Account. Within budgeting, it is possible to report on job budgets concerning quantity, cost and billing prices. It is possible to distinguish the different budget types and revisions as well as teams and planning start and end dates of the tasks on the budget lines. It is also possible to report on the task progress related to job budgets. The universe also contains a variety of specific measures in specific currency types.

The measure objects can be grouped and restricted by a variety of dimensions including company, customer, vendor, job, activity, task, employee, dates, and the ten standard dimensions.

Chapter 28

General Ledger Universe

The General Ledger universe contains objects for reporting on financial figures in Maconomy. It is possible to report on both actuals and budget figures. The universe is centered on core facts on the lowest transactional level (**Finance Entries** and **Budget Entries**).

By combining with dimension on fiscal period and the type of account, it is possible to report on summations like opening and closing balances and movements by fiscal period or other dimensions.

It is possible to organize accounts and figures according to the **Reporting Structures** set up in Maconomy. Assuming a dedicated reporting structure, it is also possible to report on financial key figures like **Gross Profit**, **Net Profit**, **EBITDA**, **EBIT**, **Cashflow** and so on. Thereby, the universe can both be used as foundation for creating some of the classical reports like balance sheets, profit and loss, and trial balances, deeper analysis or dashboarding.

The measure objects can be grouped and restricted by a variety of dimensions including company, account, customer, vendor, job, employee, fiscal periods, dates and the ten standard dimensions.

Furthermore, finance and budget figures can be reported on through local charts of accounts.

28.1 Prerequisites

In order to fully benefit from the Finance universe, Maconomy should be set up such that:

- The necessary dimension groupings should be created and setup correctly.
- The key metrics dimension grouping should designate which accounts belong to which financial key metric levels like **Gross Profit**, **Net Profit**, **EBITDA**, etc.

Chapter 29

Accounts Receivable Universe

The Accounts Receivable universe provides objects for reporting on customer transactions like invoices, payments and reconciliations; thereby, also balances outstanding on customers. It is possible to report on the balance of each entry, provision, cash discount and customer payment variance. The figures can be distinguished or restricted by entry date, due date or similar in order to categorize outstanding amounts or other amounts into aging buckets.

The measure objects can be grouped and restricted by a variety of dimensions including company, customer, job, employee, dates, and the ten standard dimensions.

Chapter 30

Accounts Payable Universe

The Accounts Payable universe provides objects for reporting on vendor transactions like vendor invoices, payments and reconciliations; thereby, also balances outstanding on vendors. It is possible to report on the balance of each entry, invoiced amounts, and vendor payment variance. The figures can be distinguished or restricted by entry date, due date or similar in order to categorize outstanding amounts or other amounts into aging buckets.

The measure objects can be grouped and restricted by a variety of dimensions including company, vendor, job, employee, dates, and the ten standard dimensions.

Chapter 31

Human Resources Universe

The Human Resources universe provides objects for reporting on employee-oriented aspects in Maconomy. It is possible to report on development aspects like courses, skills, employee positions and evaluations. It is possible to report on the financial aspects like stock options, agreements and exercisings. It is possible to report on company properties including leasings and costs; e.g. for company cars. It is also possible to report on recruitment processes and event flows related, as well as vacancies.

The measure objects can be grouped and restricted by a variety of dimensions including company, employee, employee category, dates and the ten standard dimensions.

Chapter 32

Contact Management Universe

The Contact Management universe provides objects for reporting on opportunities—i.e. sales pipeline—and the event and event flows related. It is possible to report on the individual opportunities and the contact companies and information associated. It is possible to categorize opportunity into won and lost as well as distribute over quarters, half-years or other timeline dimensions. It is possible to associate data about the events and event flows related to opportunities. It is also possible to report on the probability with which the sales prices are weighted in Maconomy.

The measure objects can be grouped and restricted by a variety of dimensions including contact company, contact person, campaign, planning dates, and the ten standard dimensions.

Index

Symbols

A

B

C

D

E

F

G

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

Y

Z

