OneStop Reporting:

Advanced period formulas

Introduction

In **Report Designer** you have the possibility to use predefined period functions during the report building. In cases where these features do not meet your needs, you can use advanced period formulas to determine the year, period(s) or day(s) in the report. This guide explains how to use such formulas in Report Designer.

Advanced period formulas

When you work on advanced period formulas, you often have to relate to two variables in each formula.

Example:

```
{PeriodCalc.GetPeriod(Parameter).AddYears(X).WholeYear}
```

Parameter is the period parameter created in the report, and *X* is the variable that determines the number of years, months/periods and days that the formula should add or subtract.

(In some cases, there will be a third variable, represented by Y in the example with a rolling interval further down in this document.)

NOTE: Advanced period functions are calculations based on the parameter value that is selected when the report is run. Due to this, users can only select a single period to run the report for when advanced and the predefined period functions are used.

Figure 1 shows where to find the parameter for the period.

1. In the **Layout Editor**, click the group to enter the expansion.

2. Click the **Lookup** button to do a lookup.

In the **Lookup** dialog, you can see the name of the parameter (marked in blue).

3. Copy the text within {}.

4. Paste this in to replace the text *Parameter* in the formulas described in this document.

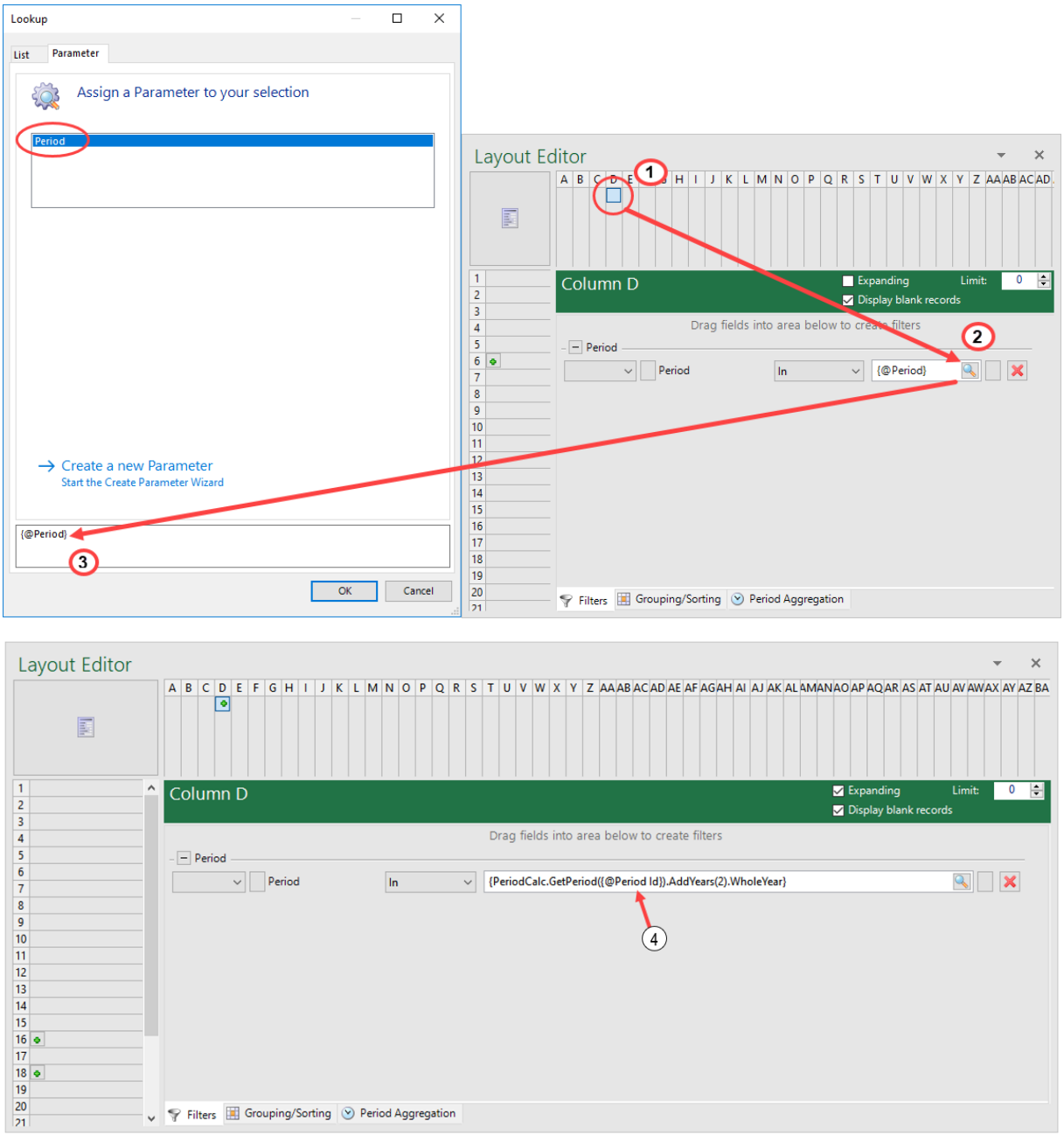

Figure 1 Finding the parameter for the period and using in in an advanced period formula

Period formulas based on a monthly ID (e.g. Period=201801 for January 2018)

The table below shows various period formulas where you can replace (`Parameter`) with (`PostingPeriods`)[1] and replace the variable X with the number of years/months you want to add or subtract. The formula element *Extend* determines how many months to add to an interval. More examples of built in period functions will follow in the section *Overview of the built-in period functions in Report Designer* below.

Period function	Period formula
Whole year in year X	`{PeriodCalc.GetPeriod(Parameter).AddYears(X).WholeYear}`
First period in year X	`{PeriodCalc.GetPeriod(Parameter).AddYears(X).YearStart}`
Last period in year X	`{PeriodCalc.GetPeriod(Parameter).AddYears(X).YearEnd}`
Q1 of year X	`{PeriodCalc.GetPeriod(Parameter).AddYears(X).YearStart.Extend(2)}`
Q2-Q4 in year X	`{PeriodCalc.GetPeriod(Parameter).AddYears(X).YearStart.SetIndex(X).Extend(2)}`
Remaining periods in year X	`{PeriodCalc.GetPeriod(Parameter).AddYears(X).Add(1)}:{PeriodCalc.GetPeriod(Parameter).AddYears(X).YearEnd}`
This period year X	`{PeriodCalc.GetPeriod(Parameter).AddYears(X)}`
Year to date year X	`{PeriodCalc.FiscYearToDate(X).AddYears(X)}`
Rolling from X to Y	`{PeriodCalc.GetPeriod(Parameter).Add(X).Extend(Y)}`

[1]The term used varies between ERP systems. Look for the period parameter, such as Periods, Posting Periods, Accounting Periods, etc. in the **Layout Editor**. (See Figure 1 above for where to look.)

Example of formula for rolling interval from X to Y

If you want to show 2 years of rolling history, then you must first subtract 1 month (from the period parameter you select when you run the report), and then extend (Extend) with 23 months.

Here, X gets the value -1 and Y gets the value -23.

The formula would like this:

```
{PeriodCalc.GetPeriod(PostingPeriods).Add(-1).Extend(-23)}
```

Period Formulas based on date ID (format dd.mm.yyyy – e.g. 01.01.2019)

The table below shows various period formulas where you can replace (DateParameter) with a date parameter[2] and replace the variable X with years/months you want to add or subtract. You find the parameter in the same way as in the example above. The only difference is that you have a period dimension based on day level, for example posted date.

Date parameter	Period formula
Year	'{DateParameter.AddYears(X).SqlDate}'
Months	'{DateParameter.AddMonths(X).SqlDate}'
Days	'{DateParameter.AddDays(X).SqlDate}'
Day Y, in year X	'{DateParameter.AddYears(X).AddDays(Y).SqlDate}'

NOTE: When the period formula is based on a date, as opposed to a period (month), you must include ' before and after the formula:
'{DateParameter.AddYears(X).SqlDate}'

[2]The term used varies between ERP systems. Look for date parameters such as Due Date, Transaction Date, etc.

Example of formula for day Y in year X

In this example, you want to find yesterday last year. You can do this by replacing -1 Y with -1. The formula will then subtract 1 year minus 1 day for the date you set when you run the report.

'{DateParameter.AddYears(-1).AddDays(-1).SqlDate}'

Example of formula with Due Date

In this example, you want to show invoices that are due in the next 30 days, based on a date parameter, @DueDate. You can do this by using this formula:

'{@DueDate.AddDays(30).SqlDate}'

NOTE: You can also use today's date as a starting point (and not use a date parameter):

'{PeriodCalc.Now.SqlDate}'

Overview of the built-in period functions in Report Designer

Name	Display Name	Expression
ALL2Yago	All Year 2 Years Ago	`{PeriodCalc.GetPeriod([]).AddYears(-2).WholeYear}`
ALL2Yfor	All Year 2 Years Forward	`{PeriodCalc.GetPeriod([]).AddYears(2).WholeYear}`
All3YAgo	All Year 3 Years Ago	`{PeriodCalc.GetPeriod([]).AddYears(-3).WholeYear}`
All3YFor	All Year 3 Years Forward	`{PeriodCalc.GetPeriod([]).AddYears(3).WholeYear}`
AllTimeTD	All Time to Date	`{PeriodCalc.GetPeriod([0])}:{PeriodCalc.GetPeriod([])}`
FPLY	First Period Last Year	`{PeriodCalc.GetPeriod([]).AddYears(-1).YearStart}`
FPNY	First Period Next Year	`{PeriodCalc.GetPeriod([]).AddYears(1).YearStart}`
FPTY	First Period This Year	`{PeriodCalc.GetPeriod([]).YearStart}`
LPLY	Last Period Last Year	`{PeriodCalc.GetPeriod([]).AddYears(-1).YearEnd}`
LPNY	Last Period Next Year	`{PeriodCalc.GetPeriod([]).AddYears(1).YearEnd}`
LPTY	Last Period This Year	`{PeriodCalc.GetPeriod([]).YearEnd}`
LY	Last Year (all)	`{PeriodCalc.GetPeriod([]).AddYears(-1).WholeYear}`
LYTD	Last year to date	`{PeriodCalc.LastYearToDate([])}`
NY	Next Year (all)	`{PeriodCalc.GetPeriod([]).AddYears(1).WholeYear}`
Q1LY	Q1 Last Year	`{PeriodCalc.GetPeriod([]).AddYears(-1).YearStart.Extend(2)}`
Q1TY	Q1 This Year	`{PeriodCalc.GetPeriod([]).YearStart.Extend(2)}`
Q2LY	Q2 Last Year	`{PeriodCalc.GetPeriod([]).AddYears(-1).SetIndex(4).Extend(2)}`
Q2TY	Q2 This Year	`{PeriodCalc.GetPeriod([]).SetIndex(4).Extend(2)}`
Q3LY	Q3 Last Year	`{PeriodCalc.GetPeriod([]).AddYears(-1).SetIndex(7).Extend(2)}`
Q3TY	Q3 This Year	`{PeriodCalc.GetPeriod([]).SetIndex(7).Extend(2)}`
Q4LY	Q4 Last Year	`{PeriodCalc.GetPeriod([]).AddYears(-1).SetIndex(10)}:{PeriodCalc.GetPeriod([]).AddYears(-1).YearEnd}`
Q4TY	Q4 This Year	`{PeriodCalc.GetPeriod([]).SetIndex(10)}:{PeriodCalc.GetPeriod([]).YearEnd}`
RB12M	12 Month Rolling (-1 -> -12)	`{PeriodCalc.GetPeriod([]).Add(-1).Extend(-11)}`
RBC12M	12Month Rolling Current (0 > -12)	`{PeriodCalc.GetPeriod([]).Add(0).Extend(-11)}`
RF12M	12 Months Rolling (+1 -> +12)	`{PeriodCalc.GetPeriod([]).Add(1).Extend(11)}`
RF3M1	3 Months Rolling (+1 -> +3)	`{PeriodCalc.GetPeriod([]).Add(1).Extend(2)}`
RF3M2	3 Months Rolling (+4 -> +6)	`{PeriodCalc.GetPeriod([]).Add(4).Extend(2)}`
RF3M3	3 Months Rolling (+7 -> +9)	`{PeriodCalc.GetPeriod([]).Add(7).Extend(2)}`
RF3M4	3 Months Rolling (+10 -> +12)	`{PeriodCalc.GetPeriod([]).Add(10).Extend(2)}`

Name	Display Name	Expression
RFC12M	12 Months Rolling Current (0 > 12)	`{PeriodCalc.GetPeriod([]).Add(0).Extend(11)}`
RPLY	Remaining Periods Last Year	`{PeriodCalc.GetPeriod([]).AddYears(-1).Add(1)}: {PeriodCalc.GetPeriod([]).AddYears(-1).YearEnd}`
RPTY	Remaining Periods This Year	`{PeriodCalc.GetPeriod([]).Add(1)}:{PeriodCalc.GetPeriod([]).YearEnd}`
TPLY	This Period Last Year	`{PeriodCalc.GetPeriod([]).AddYears(-1)}`
TPNY	This Period Next Year	`{PeriodCalc.GetPeriod([]).AddYears(1)}`
TPTY	This Period This Year	`{PeriodCalc.GetPeriod([])}`
TYALL	This Year (all)	`{PeriodCalc.GetPeriod([]).WholeYear`
YTD	Year to Date	`{PeriodCalc.FiscYearToDate([])}`