KERN

Allevo & Excel

Anwenderhandbuch

Allevo 4.3

Allevo Junan - Using SAP & Excel efficiently and effectively

Allevo Junan combines Excel in SAP, integrating a wide range of business processes into the leading standard business software.

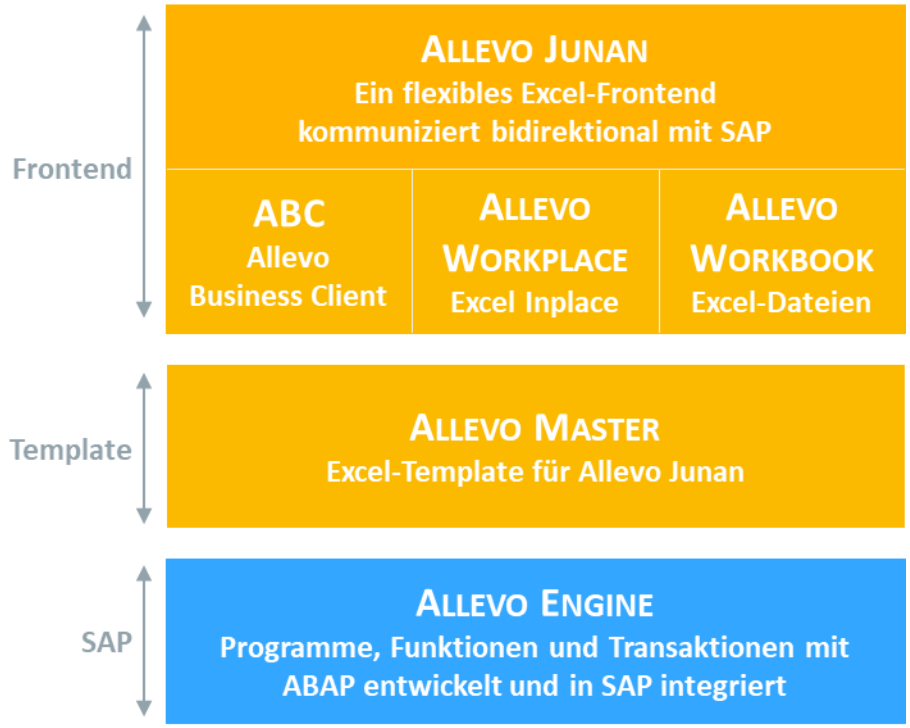
In this manual, we deal exclusively with the Excel side of Allevo Junan. You'll learn the basic structure of the Allevo Master. From there, we'll develop cooking recipes that you can use to model Allevo Master according to your requirements.

Content - Overview

1 Allevo Junan (overview)

Allevo Junan is a flexible Excel front-end, it reads and writes data from SAP into a user interface preconfigured in Excel.

We call these **Allevo Master.**



Allevo Junan

There are three forms for using the Allevo Master:

Allevo Workplace

The Excel front end embedded in SAP (SAP technology: Excel-Inplace):

In this application form, Excel is integrated into the transactions of the SAP system.

The user first logs on to the system via the SAP Logon: various specific Allevo transactions are available there, for example, to enter the selection parameters required for planning. When the Allevo master is called up, Excel also opens as part of the SAP application (see detailed description in the Allevo SAP manual).

Depending on the Allevo transaction, the master is called in *MultiObject*, *MultiPage* or *Reporting* mode.

Allevo Workbook

A stand-alone Excel document that can be edited independently of SAP. This can then be edited offline by employees without SAP access. The data can then be transferred to SAP. We call this workflow offline planning.

Allevo Business Client (ABC)

In this type of application, the Allevo master is started via an independent application from the planner's Windows workstation (e.g. via a desktop icon). A selection window (panel) appears with all the information required to call up an Allevo master: e.g. information on the layout and the object data. The panel thus has

the same functionalities as the Allevo transactions in the SAP system. Logging on to the SAP system also takes place from here. Via the panel, the planner sees all the layouts set up for Allevo in the SAP system and can select accordingly: Excel is started and opens the selected master. A stored addin takes over the entire data exchange with the SAP system from there.

Alternatively, the Allevo Business Client can also be started directly via the Allevo Master (e.g. by double-clicking on the file or also via a portal). This is also how offline files are integrated for data transfer to SAP.

The Allevo Business Client is primarily intended as an alternative form of data entry. The customizing associated with Allevo (constant in the layouts, global constants...) continues to be carried out via the Allevo transactions on the SAP side.

The special features of the ABC form of use are described in the Allevo Business Client manual.

Regardless of the application form, the same Allevo Master can be used in all three cases.

Recommendation

Use the inplace variant if the planners are used to working directly in the SAP system and special features, such as jumping to the document, are important. With this form of application, no special installation of software on the planner's workstation is required.

With the ABC variant, all Excel properties are available, even those that are not supported by the SAP GUI depending on the Excel version (e.g. print preview or status bar functions). This variant also has advantages if direct accesses from the planning file to other files of the Microsoft environment are desired.

Note:	In principle, both processes can be operated in parallel. The program logic in the Allevo master independently recognizes which mode is being used. A user-specific login is required for both modes. As a result, all SAP authorizations and consistency checks are adhered to even when called up via ABC.

Allevo formulas

The Allevo-Master can be operated in two different formulary concepts:

- With the MultiObject form, all objects are listed as an Excel table, whereby different object types (e.g. cost centers and orders) can be colorfully mixed. The structure of this list can be fixed or dynamic.
- When executed in MultiPage mode, the main worksheet is duplicated as many times as objects have been selected. After that, the data for all selected objects are available in the same Excel file: For this reason alone, this format is preferred, especially for planning with offline files.

The Allevo Master sample template supplied can be used for all the above-mentioned form concepts (MultiObject and MultiPage).

Allevo Excel Basic Rules

When designing the Allevo Master, please observe the following rules:

- In cells that contain key values or that are relevant for navigation, Excel must not return errors (e.g. #NV or *#VALUE*). These errors can occur when the cell contents are derived using formulas. We recommend catching errors using the *IF ERROR* function.

- A zero is scheduled, an empty cell is not. A complete deletion of data via Allevo is not provided due to this principle, only the reset to zero. Even if a formula returns "", the value is not passed to SAP, unlike a returned *zero*.
- Excel cells that contain a formula are not overwritten by data from SAP. Allevo assumes that the cell value is recalculated by the formula.
- The security settings must allow the execution of Kern AG macros. The Excel masters are provided with a signature of Kern AG, the associated certificate must be installed if necessary.
- Instead of individual formatting, defined cell formats are always preferable. A number of predefined formats are available:

KernAttention	KernRead	KernWrite	KernSum01
KernHeadine1	KernRead%	KernWrite%	KernSum02
KernHeadine2	KernRead0	KernWrite0	KernSum03
	KernRead1	KernWrite1	KernSum04
	KernRead2	KernWrite2	KernSum05
	KerReadDate	KerWriteDate	KernSum06
			KernSum07
			KernSum08
			KernSum09
			KernSum10

Hervorhebung	1234,56	1234,56	1234,56
Überschrift	123456%	123456%	1234,56
Überschrift	1.235	1.235	1234,56
	1.234,6	1.234,6	1234,56
	1.234,56	1.234,56	1234,56
	18.05.1903	18.05.1903	1234,56
			1234,56
			1234,56
			1234,56
			1234,56

Predefined cell formats

Glossary

For a better overview when reading, the following notations are used in this manual:

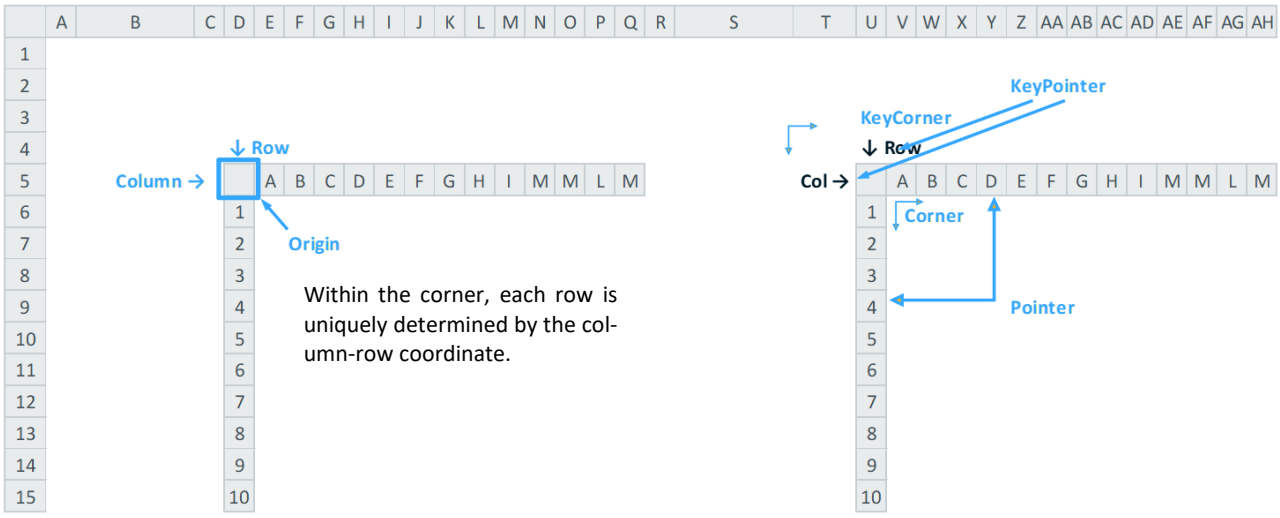
- **Table sheets** are bordered with pipes, e.g.: |Standard|, |Navigation|
- **Structured tables** are enclosed in square brackets: [StyleAreas], [SplasherCommonSettings].
- **Range names** can be found inside curly brackets: {Navigation01Row}, {TT_SUM}

2 The Corner Concept

In addition to the familiar Excel area names, Allevo uses so-called corners to define the area of effect of various functions:

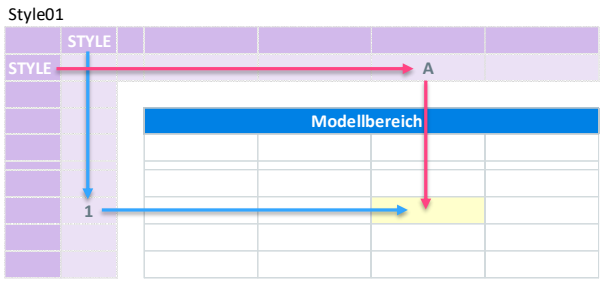
You are already using the corner concept today: The largest conceivable corner in an Excel spreadsheet is created with the two arms for the column and for the row coordinates. At the zero point, both arms cross and thus form an angle. - This is Excel's own corner.

The (invisible, but implicit) KeyPointers *Column* and *Row* indicate "this row contains the column coordinates" (KeyPointer *Column*) and "this column contains the row coordinates". We call the coordinates *pointers,* the invisible "outer" pointers we call KeyPointers.



Corner Concept in Excel

"Our" Corners use more KeyPointers than the Excel standard for its CoordinateCorner, but Allevo Corners are also always about the combination of a column and a row.



Example of an Allevo Corner

The area enclosed by the corner is called the *model area.* It contains the cells on which the functions belonging to the corner can act.

In the Corner itself we distinguish two areas:

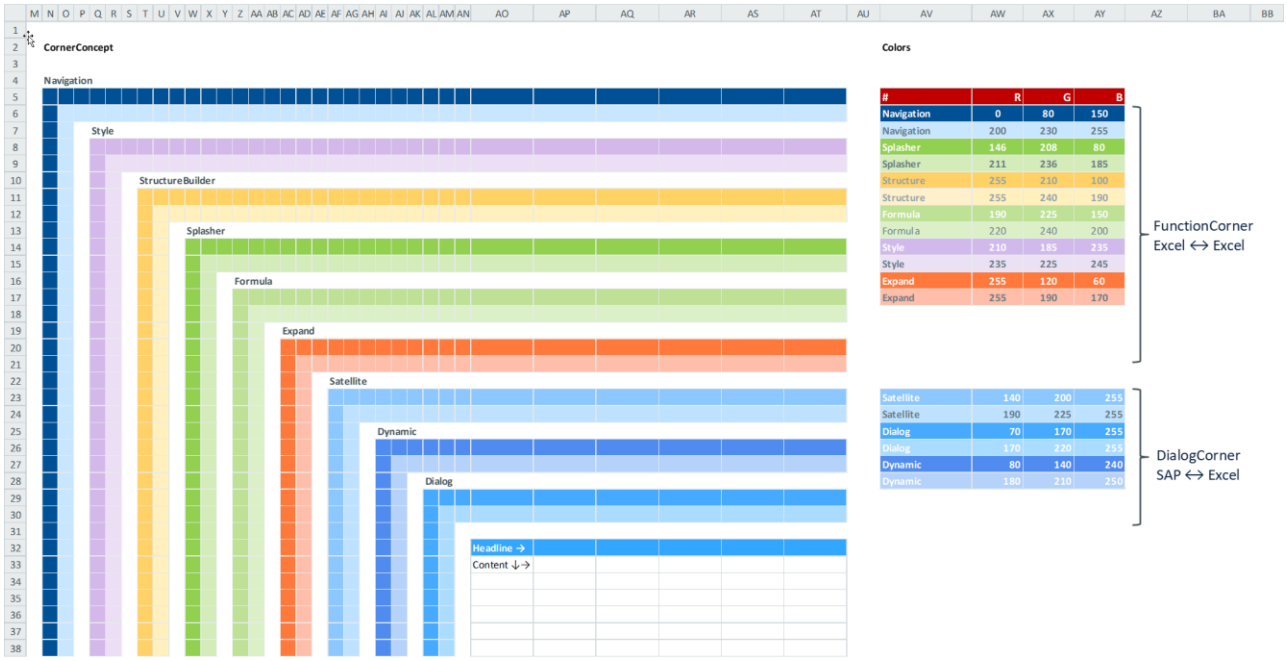
- The KeyPointers are located in the dark outer area of the corner and define the function
- The pointers are located in the bright inner area of the corner and define the area

In the above example, the KeyPointer *STYLE* together with the pointers *A* and *1 do* the following:
The yellow cell is formatted when the style function is called.
(The format for the combination of *A* and *1* is stored in a configuration table).

The axes of a corner are assigned to name ranges, which are structured according to the following scheme:

- {<CornerName><Nr>Row} for the row details runs vertically ↓
- {<CornerName> <Nr>Column} for the column details runs horizontally →.
- So in this example *Style01Column* and *Style01Row*

Depending on the requirements, you can nest as many corners as you like.



2.1 Many corners - the right solution for every occasion

Note:	It is possible to use several corners of the same module. These are then incremented in the last two digits of the name, so e.g. Formula01, Formula02, etc....

3 Structure of the master

Brief overview

The delivered sample template of the Allevo master is a standard master for all SAP object types. It consists of a mandatory basic structure and optional additional modules.

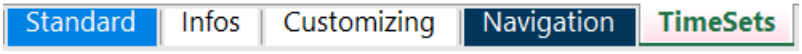
Each of these functional elements consists of the following components:

- An impact area, defined by a corner or area name (Impact Area).
- One or more tables with parameters for configuration (Settings)
- A set of control commands (Controls)
- One or more VBA or .NET modules (code)
- The basic structure additionally includes global settings and transfer tables that are available across functions.

A corresponding brief overview precedes each chapter in the manual.

Basic structure

A sheet is assigned to each functional area of the basic structure in the master:



3.1

The basic structure contains the following modules:

Standard, Info, Customizing, Navigation, TimeSets

Optional modules

With the optional modules, the basic structure can be extended by numerous functions.

- **Dialog**

  Provides communication between the Excel frontend and SAP standard tables.

- **Satellite**

  Establishes the bidirectional connection between the Excel frontend and SAP. This allows you to easily write and read out customer-specific e.g. specialized planning such as investment, marketing, travel costs etc. into your SAP system.

- **Dynamic**

  By means of the Dynamic Module, the row and column structure dynamically adapts to the object / element structure read from SAP.

- **Style**

  With the Style module you apply the principle of conditional formatting. However, you go further than the Excel standard, because you can ensure that style sheets are applied.

- **Formula:**

  Depending on your use case, you may want to assign different formulas to one or more cells.

- **Splasher**

  With this Building Block you skip the annoying problem of circular references: You specify a yearly value, the splasher distributes to months; you specifically adjust the month(s), the splasher accumulates the yearly value.

4 Standard

Your individual Allevo user interface is created on the |Standard| sheet.

It contains the table [SheetObject] and a structure of several Corners. [SheetObject] references the standard area on entry object / object type / cost accounting circle, the corners control the further posting process. [SheetObject] is located in cell *B2* above the corners.

SETCLASS	OBJECT	COMPANYCODE	DESCRIPTION
KS	1200	1000	Prod.M

4.1 SheetObject

5 Info

In the sheet [Info] there are two structured tables: [Global Information] and [Local Information].

These tables are filled with transfer parameters and information on the initial objects when they are called from SAP. These are available in Excel via name ranges.

Note:	To simplify the design of the master, this table is often stored in the master already filled with default or test values. These are then overwritten with the data from SAP at runtime.

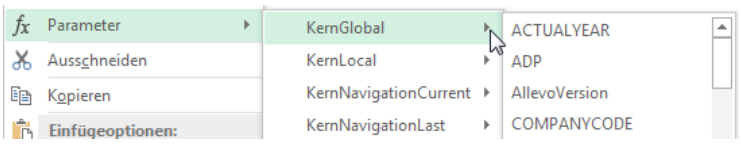
Global Information

Key	Value	Description
CONTROLLINGAREA		
OBJECT		
VERSION		
PLANYEAR	2022	GLOBAL_PLANYEAR

Global Information

The global object information always refers to the entire workbook and the initial object selected during planning entry, whereas the local object information is different in multipage mode or when using the Allevo tree for object selection. This is provided as a list in [Local Information].

Two core functions are available for accessing this object information (KernGlobal and KernLocal), which can be conveniently called via the context menu:



Inserting object information via parameter function

Of course, it is also possible to optionally refer to the Infos worksheet using ordinary Excel formulas.

The reference to the main object of a page is not always sufficient; especially if objects and object types change in the respective sheet. In this case, information on the row level or on all mentioned objects is of interest. In this case, the *Property* pointer in the dialog corner can help: it provides master data info for all objects in the Allevo master, which is otherwise only available via CUSTINFOxx.

List of available parameters

Here is a list of the currently available parameters (for both Global and Local):

Cell	Meaning
CONTROLLINGAREA	Controlling area according to selection in SAP. Note: please pay attention to correct formatting. A controlling area "0001" differs from "1" on the SAP side, which is why there is no automatic format adjustment.
OBJECT	Object number in SAP-internal representation (i.e. with leading zeros for cost centers etc.). In MultiPage mode it is the object number on the current sheet. Special case: for WBS elements OBJECT contains the external representation; as it is also required for the later data transfer between SAP and Excel. If an internal representation is required for WBS elements, MAP_FIELDS should be used (e.g. for access to data in the satellite).

VERSION	Version to be planned in, according to Allevo settings in SAP (corresponds to the version entered in the TimeSet marked as Planning Base on SAP side). Same content as for GLOBAL_VERSION.
PLANYEAR	Fiscal year according to Allevo settings in SAP (Planning Base - TimeSet). Same content as for GLOBAL_PLANYEAR.
SETCLASS	Setclass as identification of the SAP object type that is being processed: e.g. "0101" for cost centers, "0103" for orders, "0110" for WBS elements. The cell should have text format. For the data transfer to SAP alternatively the object type can be entered directly (e.g. "KS" equivalent to "0101", see also CC_OBJECTTYPE).
RESPONSIBLE	Person responsible entered in the SAP master record for the current object
PERITO	To period according to Allevo settings in SAP (Reporting-Base-TimeSet) in the column definitions). Same content as for GLOBAL_PERITO.
TEXT	Object description (short text from master record for cost center, order...). See DESCRIPTION field for associated long text.
SYSID	SAP system on which the file was created
DATE	Date of file creation
ACTUALYEAR	Year setting of the actual read definition
TRANSACTION	Allevo transaction with which the file was generated
TRANSACTIONTYPE	Allevo transaction type with which the file was generated MP: Multipage MO: Multiobject
LANGUAGE	Logon language
LOGSYS	SAP system on which the file was generated
GROUP	Standard hierarchy area, i.e. group from master record to object (only for KS,PC,GP, otherwise empty).
PROFITCENTER	Profit center to which the selected object is assigned
LSTARS	Yes/No for reading the dynamic service relationships
ADP	Yes/No for activating activity type dependent planning
USER	User who created the file or special function "User role" if constant USER_DATA is set.
LAYOUT	Allevo layout with which the file was generated
FROM	For interval selection: From value
TO	For interval selection: To value
OBJECTGROUP	Name of the group when entering via MultiPage or MOD
PROJECT	Only for WBS elements: Assigned project
STATUS	Allevo planning status for the current object
DESCRIPTION	Detailed object description (long text from master record for object, available for KS, PC and BP, otherwise ID of assigned cost centers). See TEXT field for associated short text.
ALLEVOVERSION	Allevo version number in SAP
COMPANYCODE	Company code of the planning object
GROUPNAME	For group selection: description of the selected group
FCODE	Technical function code from SAP

REPRESENTATIVE	Representative from master record
OBJECTTYPE	Identification of SAP object type as an alternative to CC_SETCL (Allevo supports KS, BP, OR, PC, BP, PR). See also GLOBAL_OBJECTTYPE with the equivalent specification for the representative object.
COSTCENTER	Assigned cost center of the object
CUSTINFO1-10	The fields CC_CUSTINFO1 to CC_CUSTINFO10 can display any additional information about the current object, e.g. stored in the respective SAP master record; but also additional information, e.g. the name of an assigned 1:n group or an indicator, if the reporting mode is active (see detailed F1 documentation for constant MAP_FIELDxx). For applications in MO mode all info can also be provided on row level (see constant READ_ELEMENT_DATA).
SHEETCOMMENT	Important: currently still to be transferred via namespace {CC_COMMENT}. Here you can save a long comment for the planning (sheet comment). This comment applies generally to the object (cost center, order or WBS element). It can contain more than 255 characters. The text can be called up and edited centrally via the Allevo menu Satellite tables (stored in table /KERN/IPPLTEXT).
ISOLANGUAGE	Logon language of the SAP system under which the file was created
CURRCONTROLLINGAREA	Currency of the controlling area (according to Reporting-Base-TimeSet)
CURROBJECT	Object currency (according to entry in the master record of the current object)
TREEVIEW	Yes/No for active tree display
CURRATE	Conversion rate from object currency to controlling area currency, for object type PC: exchange rate company code currency to local currency. Read from SAP customizing table TKA07, or T895PCA (for details see F1 doc for constant EXCH_RATE_PARAM)
OFFLINEPROCESSING	Information whether the file was created via offline export: X or empty
OBJECT1	Like OBJECT, but in external representation (e.g. without leading zeros). For WBS elements it is the same content as in OBJECT, because in this case the external representation is already stored there.
FCMODE	Forecast Mode: If planned year = actual year: '1' otherwise '0'.

Further use: Consistency check in offline mode	If planning is to be continued with an Allevo offline file, the local parameters OBJECT, VERSION, YEAR and PERITO are compared with the information entered in the Allevo start screen. If the file does not match, there is a conflict. The file will not be opened.

6 Customizing

The |Customizing| has two tasks: Flow control and global settings.

It contains the control table [Customizing].

UserExits

Rows with **UserExit** in the *Process* column are used for flow control,

Rows with **Settings** define global settings:

Customizing

Process	SubProcess	Parameter	Key	Description	Value
UserExit	Macro	OPEN_IN_SAP			
UserExit	View	AFT_READ			Standard01
UserExit	Macro	AFT_READ	2	ExecuteFormulas	FormulaActivate
UserExit	Macro	AFT_READ	1	Sets and Starts Sum-Formulas	startSum
UserExit	View	WB_OPEN			Standard01
UserExit	Macro	WB_OPEN			
UserExit	Macro	AFT_LEAD_SAT			
UserExit	Macro	AFT_PLAN			
UserExit	Macro	AFT_PLAN_S			
UserExit	Macro	AFT_READ_S			
UserExit	Macro	AFT_SHEET_GEN			

UserExits in the Allevo Customizing sheet

UserExits are jumps to specific events that can be used to call macros or views (navigation).

Thus, in the above example, the view *Standard01* defined in the navigation is called after reading the data from SAP (*AFT_READ*). If Excel formulas are also to be calculated after the execution of events, the execution of the *RefreshCalculation* macro may be required.

If several macros are to be called for one parameter, a new row is inserted for this purpose: In this case, the order of processing is from top to bottom. If you want to change the order, you can do this by making an entry in the "Key" column.

UserExit	View	AFT_READ			Standard01
UserExit	Macro	AFT_READ	2	ExecuteFormulas	FormulaActivate
UserExit	Macro	AFT_READ	1	Sets and Starts Sum-Formulas	startSum

UserExits order

UserExit	Meaning
AFT_DYNAMIC	Called after all (new) dynamics have been transferred to Excel. Can be used e.g. for *startSum* in the dynamics. Is only called if at least 1 dynamic corner is used.
AFT_LEAD_SAT	Relevant for satellites for which constant READ_ORDER_SAT is active: Executed AFTER reading a satellite, but BEFORE reading the plan data (i.e. between BEF_READ and AFT_READ). The satellite data can thus be the basis for further read functions (sequence applies per sheet in MultiPage mode).
AFT_PLAN	After planning all pages. Is always called after AFT_PLAN_S.
AFT_PLAN_S	After the complete planning of a page.
AFT_RD_SAT	After reading all satellites/the satellite. Called only for the additional functions "Read satellite" and "FP-Read" (function codes "EXIT1" & "FP_READ").
AFT_READ	After reading all pages. Is always called after AFT_READ_S.
AFT_READ_S	After reading a page in its entirety.

AFT_SHEET_GEN	After generating each worksheet Excel internal
AFT_SHEETS_GEN	After calling the macro *genMulti*. After generating the MultiPage worksheets once at the end.
AFT_START_SAT	Comes after the transfer of the MOD satellites (and thus also after Sat00). Comes only if also activated in Excel (keyword EXPUSEREXIT). Event is Excel-only -- no BAdI call at this point.
AFT_TT	Always executed after updating the data on the total sheet.
AFT_WR_SAT	After saving all satellites/the satellite. Called only with the additional function "Save satellite" (function code "EXIT2").
BEF_PLAN	Before handing over the plan data
BEF_PLAN_S	(with _S after each worksheet)
BEF_READ	Before reading plan data from SAP as a whole.
BEF_READ_S	(with _S after each worksheet)
BEF_RD_SAT	Before reading satellite data in SAP
BEF_WR_SAT	Before writing satellite data to SAP
CLOSE_IN_SAP	Called directly before closing the master.
MOD_TEMPLATE	Between OPEN_IN_SAP and AFT_SHEET_GEN, before setting the header data, only for MOD layouts.
OPEN_IN_SAP	Open master in SAP: is executed when the master is started in the planning environment (i.e. SAP-Inplace or via ABC). Is called only once.
WB_OPEN	Opening the master outside SAP
WB_CLOSE	Closing the master outside SAP

Settings

Global settings are available on the |Customizing| worksheet

Settings	Master adjustment after opening in SAP/Excel	Password		No sheet and workbook protection if empty	Allevo
Settings	Master adjustment after opening in SAP/Excel	ShowCustomizing		Show configuration and customizing sheets on Allevo	0
Settings	Master adjustment after opening in SAP/Excel	HideRibbon		Hide Allevo ribbon	
Settings	Master adjustment after opening in SAP/Excel	HideFormBar		Hide Form Bar	
Settings	Master adjustment after opening in SAP/Excel	LockShowAll			
Settings	Settings for Multi and MultiObject	MOWO		MO with assigned objects	
Settings	Settings for Multi and MultiObject	NamingRule		Naming rule for Multi Page	
Settings	Settings for Multi and MultiObject	CopyMultiSheet		Protect digital signature when creating multi sheets	
Settings	Settings for Multi and MultiObject	FISH		MO for ProfitCenter with fixed Sheets	
Settings	Settings for data format in Excel	FirstSheet		Name of first sheet	
Settings	Settings for data format in Excel	WriteTechnicalZero		Delete-Option: Set Zero for definend Formats/Styles instead deleting	

Customizing Settings

Settings	Meaning
Password	An entry in "Password" protects sheets and workbooks as soon as the Master Inplace or ABC has been called. The protection is also valid in offline mode and always depending on whether read objects are recognized.

	This recognition takes place on the basis of an entry >0 in the column "Status" of the table [LocalInformation] on the sheet [Info].	
ShowCustomizing	Empty	The master is displayed as saved
	0 or FALSE	The spreadsheet |Customizing| and all configuration sheets are hidden when the workbook is opened
	1 or TRUE	all spreadsheets are displayed
HideRibbon	Controls the display of the Excel ribbon. The entry is evaluated when opening the Inplace workbook or via the ABC.	
HideTabs	Controls the display of the Excel sheet tab. The entry is evaluated when opening the Inplace workbook or via the ABC.	
HideFormBar	Controls the display of the Excel edit bar. The entry is evaluated when opening the Inplace workbook or via the ABC.	
LockShowAll	Suppresses the display of control information for rows and columns in the Allevo master "Ctrl+Shift+A" is then no longer possible. **We recommend setting this parameter whenever sensitive data is handled.**	
MOWO	This can be used to activate planning with variable sheet assignment (1 or TRUE)	
NamingRule	Defines a naming rule for the sheets in MultiPage or MOWO. Without specific naming, the default defaults apply (i.e. *IPP_xxxxx*).	
CopyMultiSheet	In Excel environments with high security settings, the Allevo master is usually provided with a code signature. Due to Excel-specific properties, this signature can be lost when Allevo attempts to create the required sheets in MultiPage mode. In this case, copying should be done via the "CopyMultiSheet" method, i.e. set to TRUE.	
FirstSheet	If a sheet name is entered at "FirstSheet", this sheet will be activated when opening and after reading/planning.	
WriteTechnicalZero	Activates a function to automatically enter a 0 in the respective cell when deleting cell contents (via keys such as Del, Backspace) or for all empty cells in certain areas. This transfers the cell content to SAP and also resets the plan value. Optionally, a confirmation can be requested from the planner (popup before setting the zero values).	
FISH	(Fishbase = Fixsheet) MultiObject for profit centers: Master with fixed sheets for different company codes with the same object.	
AcitvateReportKit	Activates the optional Building Block *ReportingKit*	
StatKey	Shortcut for the optional Building Block *Statistical Overview*	

Note:	Other settings are used to configure certain optional Building Blocks. They are described in the corresponding chapter.

7 Navigation

Navigation01

	SHOWROW								
SHOWCOLUMN									

With this module you control the navigation and configure the different views of the user interface.

For each view a button is created in the ribbon, this is displayed via the menu item *Navigation.*

When the user clicks on a menu item, the navigation hides or shows columns and rows and the user gets the assigned section.

Columns to the left of the NavigationCorner are hidden, column A is shown and the width is set to 1.

Rows above the NavigationCorner are hidden. If the "SheetObject" table exists on the sheet, the row below it is shown. If the table does not exist, the first row is shown. In both cases, the height is set to 8.

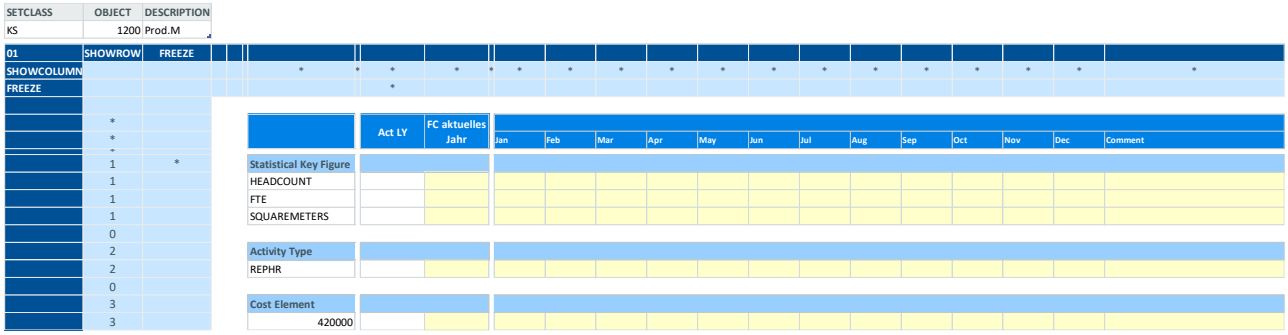
Brief overview

Area	Feature	Info
Code		
VBA	basNavigation	
	clsNavigationView	
Settings		
Configuration sheet	Navigation	
Setting tables	NavigationSheets	
	NavigationViews	
	NavigationCustomButton	
	NavigationHistory	
Controls		
Macro		
Cado		
UserExits		
Impact Area		
Corner	NavigationXX	
KeyPointer	SHOW	Controls the display of the row (SHOWROW was used up to 4.1)
	SHOW	Controls the display of the column (SHOWCOLUMN was used up to 4.1)

	FREEZE	Defines the first scrollable row/column

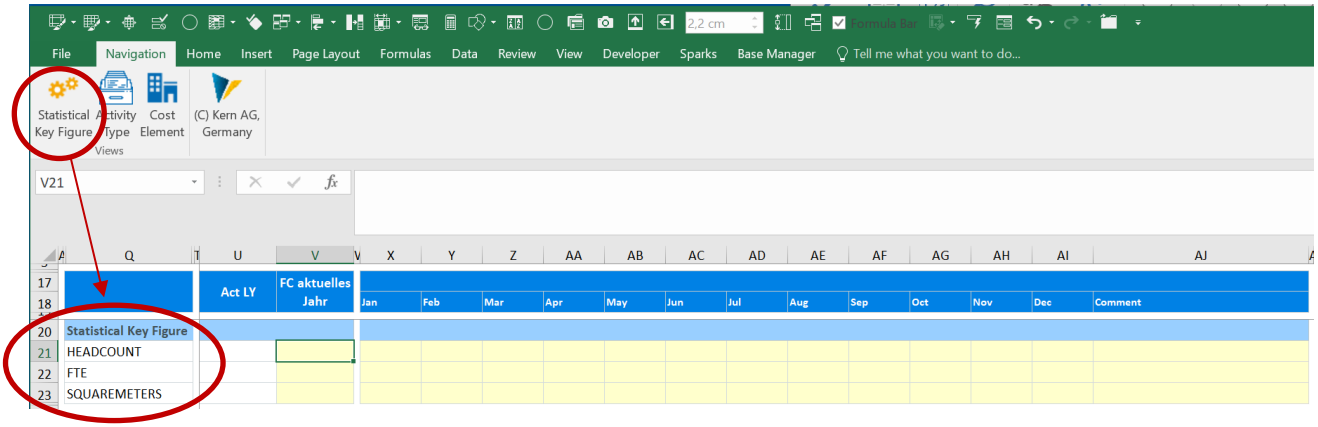
7.1 Example

1. Structure of the worksheet with NavigationCorner:



NavigationBefore

2. Selecting the *StatisticalKeyFigures* view in the navigation ribbon calls a view that displays only the rows containing statistical key figures:

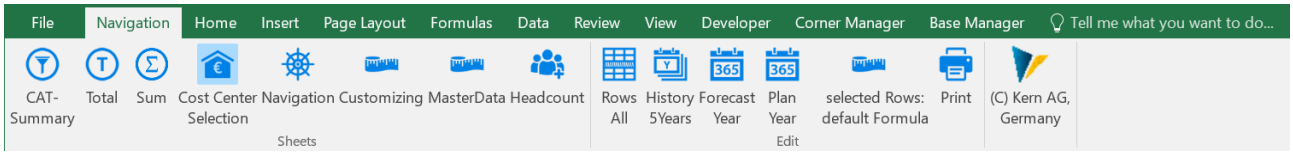


NavigationAfter

7.2 Ribbon (Ribbon)

Navigation within the Allevo Master is mostly triggered by commands in the *Navigation* ribbon: by clicking on an icon, the user accesses special views of the planning form.

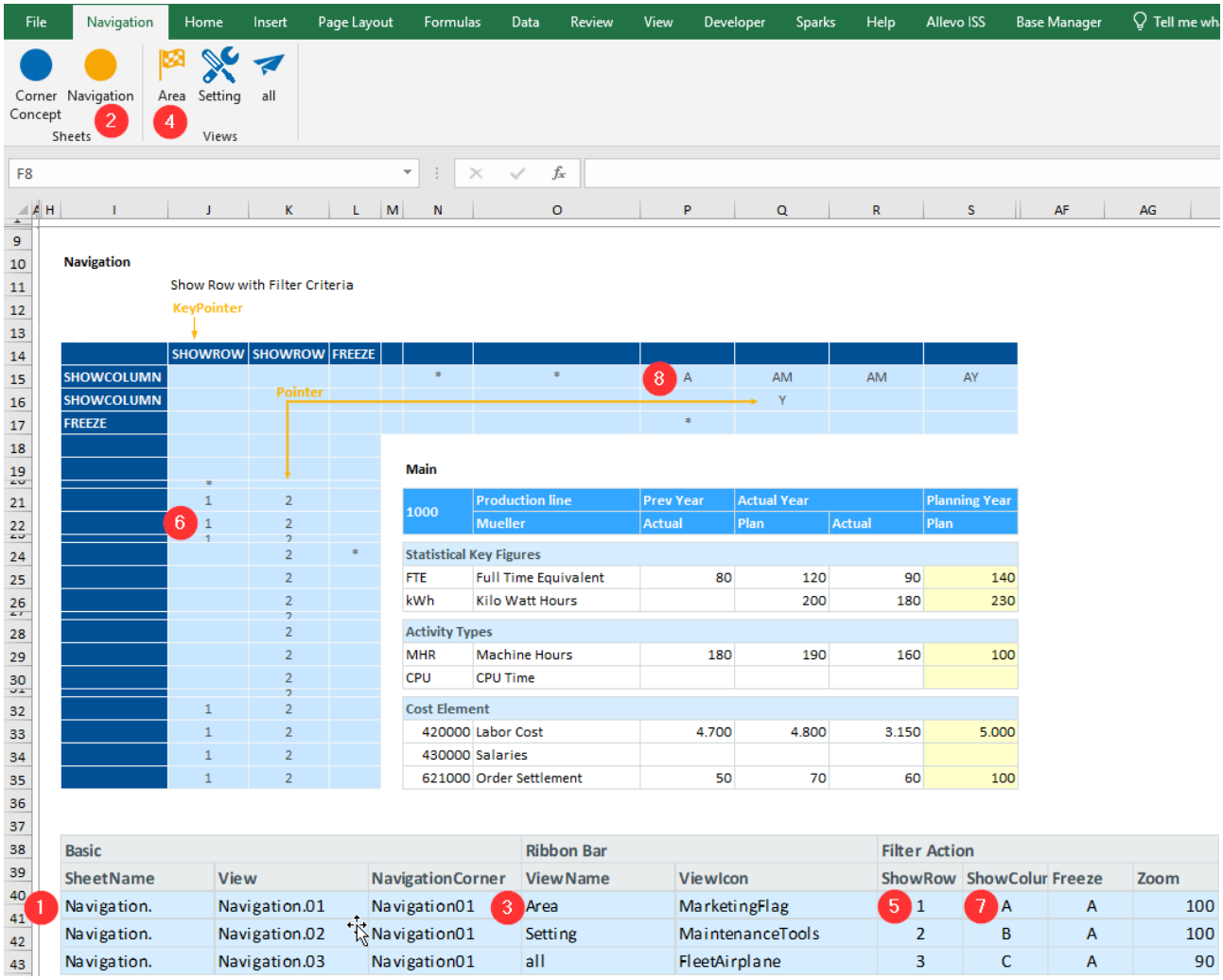
Navigation can also be triggered via UserExits, the context menu, hyperlinks or the SheetChange event (Autofilter option).



Example of a ribbon with different elements

7.3 Basic configuration:

Navigation is configured by making entries in control tables and setting **KeyPointers** and **Pointers** in the Corner: In the following example, an *Area* view is created for the sheet |Navigation|, which displays the rows marked with *1* and the columns marked with *A.*



The pairs ①&② | ③&④ | ⑤&⑥ | ⑦&⑧ belong together.

①&② The entry in the control table in the spread sheet |Navigation| creates the Sheet menu item. This allows you to navigate from spreadsheet to spreadsheet.

③&④ The *ViewName* creates the menu item for navigation within a spreadsheet.

⑤&⑥ *ShowRow specifies* that when you click on the menu item ④, all rows are hidden that do NOT contain the digit 1. - Note the KeyPointer *ShowRow* in cell J14: The NavigationCorner searches in all cells below the KeyPointer *ShowRow* for the digit *1*. - In addition, all rows remain visible in which (below the KeyPointer *ShowRow*) the character * (= wildcard) is located.

⑦&⑧ *ShowColumn* specifies that all columns are hidden that do NOT contain the pointers *A* or ***. To do this, NavigationCorner searches to the right of the two *ShowColumn* KeyPointers (cells I15:I16) for the two pointers *A* & ***.

7.4 Configuration tables

NavigationViews

In the *NavigationViews* table the individual views are configured, each row corresponds to one view:

Views

Basic				Ribbon Bar			Filter Action				Macro		View Specials					
Count	SheetName	View	NavigationCorn	ViewName	ViewName2	ViewIcon	ShowRow	ShowColu	Freeze	Zoom	MacroBeforeFi	MacroAfterFilt	AutoFilter	HyperLink	ColumnWidth	RowHeight	SizeUnit	Tooltip
01	Standard	Standard01	Navigation01	active		Filter-Standard	1	A	A	100								Start your planning here
02	Standard	Standard02	Navigation01	all		Euro	2	A	A	100								
03	Standard	Standard03	Navigation01	Forecast		ReportingGraphArrow	1	B	A	100								
04	Standard	Standard04	Navigation01	Plan		CustomBagTool	1	C	A	100								
05	Standard	Standard05	Navigation01	Kennzahlen		Chart	3	C	A	100								
01	Invest	Invest01	Navigation01	neu		DetailsHouse	1	A	A	100								Plan your assets here
02	Invest	Invest02	Navigation01	alles		MaintenanceTool	1	B	B	100								Check the orders allocated in your asset planning

NavigationViews

With these properties, all rows with a *1* (or ***) and** all columns with an *A (or **) are* shown in the *Main01* view with the *Main* label on the *Allevo sheet in* the Navitation01 corner.
The window is fixed in the position marked M (KeyPointer *FREEZE*). Zoom is set to 100% and the column widths/row heights are controlled in the by the pointers "WIDTH01/HEIGHT01".

Configuration options

Setting	Description
SheetName	Name of the Excel worksheet the view refers to
View	Name of the view (where the button should appear)
NavigationCorner	Corner name
ViewName	Designation on the ribbon button
ViewName2	Name of the subitem on the ribbon button
ViewIcon	Name of the icon used
ShowRow	Filter criterion for rows to be displayed, "*" always displays the row
ShowColumn	Filter criterion for columns to be displayed, "*" always displays the column
Freeze	Pointer for fixing the window
Zoom	Zoom percentage
MacroBeforeFilter	Name of the macro that should run before executing the view
MacroAfterFilter	Name of the macro which should run after execution of the view
AutoFilter	Name range that triggers the row filter as soon as a cell content changes
HyperLink	The name of the hyperlink must be the same as the name range of the cell in which it is located. The hyperlink is executed together with the view
ColumnWidth	Column widths
RowHeight	Row heights
SizeUnit	Point (PT) or pixel (PX), where on Windows 1PT = 3/4 PX. If empty, then Excel's default units are used.
Tooltip	Text that appears at the button when the cursor is over it

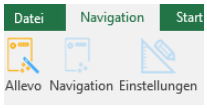
NavigationSheets

These tables control the creation of ribbons per worksheet

Sheets

Menu for Sheet			
SheetName	SheetAlias	SheetIcon	Tooltip
Allevo		CustomSheetMagicWand	Enter your planning values here
Navigation		CustomSheetMagicWand	Define your own views and buttons
Customizing	Einstellungen	DetailsPenRuler	Configure your Allevo settings. Only for Administrators

With the exemplary entries the following ribbon with the jump option to the individual spreadsheets is generated:



Setting	Description
SheetName	Name of the Excel worksheet the view refers to
SheetAlias	Name of the view (where the button should appear)
SheetIcon	Name of the icon used
Tooltip	Tooltip of the icon

NavigationCustomButtons

With the help of the CustomButtons macros buttons can be assigned in the ribbon. The necessary settings are made in the *NavigationCustomButton* table:

Custom Button

Custom Button					
SheetName	View	Icon	MacroName	Caption	Tooltip
Allevo		CustomSheetMagicWand	StartMakro	Mein Makro	Mein eigenes Makro

Setting	Description
SheetName	Name of the Excel worksheet the macro refers to
View	Name of the view the macro refers to
Icon	Name of the icon used
MacroName	Name of the called macro
Caption	Text under the icon in the ribbon
Tooltip	Tooltip of the icon

Memory setting of a navigation angle

The last performed navigation and its most significant properties can be queried via the *NavigationHistory* table by formula. The last performed navigation is stored in the *Current* column, the previous one in the *Last column*.

History

View Status		
Property	Last	Current
View	Invest01	Invest01
FilterRow	MainRow	MainRow
FilterColumn	MainColumn	MainColumn
ShowRow	1	1
ShowColumn	I	I
Freeze	A	A
Cell	B5	H14

NavigationHistory: finding the last navigation performed

Note:	The reference to the last executed view makes sense for the following constellation, for example: Button A selects row filter **1** (FilterRow = 1), Button B row filter **2** (FilterRow = 2). If button C is now used, it should keep the row filter of the previous button, regardless of whether this was button A (i.e. 1) or button B (i.e. 2).

7.5 Toggle button for showing and hiding zero rows (optional)

The *SuppressZeroRows* toggle button is an addition to the navigation and shows or hides the zero rows without the need for two separate buttons.

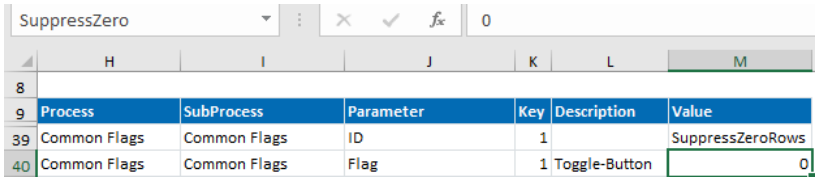
The toggle button is created as a custom button as usual, the icon *Filter* would be suitable here, but as always it is freely selectable. The macro *ShowHideZeroRows* is entered here. Since the identifier (*Caption* column) is to change at runtime - depending on the status of the toggle button - it also receives the area Z_SUPPRESS_ZERO_CAPTION.

Custom Button

Custom Button					
SheetName	**View**	**Icon**	**MacroName**	**Caption**	**Tooltip**
Standard		DetailsPenRuler	ShowHideZeroRows	ShowHideZeroRows	Shows / Hides Zero Rows

NavigationShowHideZeroRows

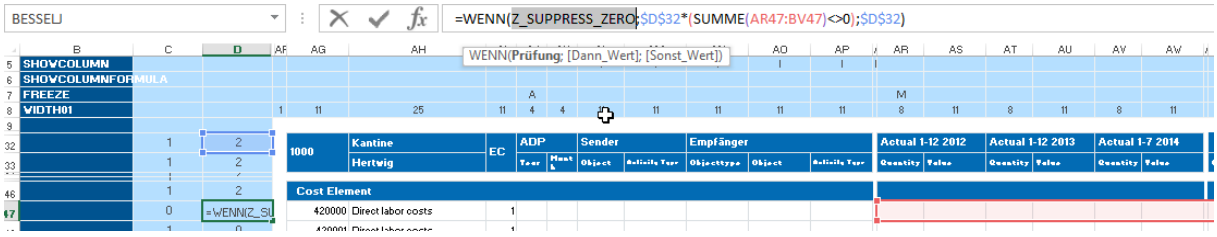
The macro sets a flag each time it is called, which is set up on the customizing sheet as follows:



Customizing setting for toggle button flag

In order to be able to access the respective value in the NavigationCorner via formula, the value cell is provided with a range (in the example, the range is called {SUPPRESS_ZERO}, but this is freely selectable).

In the NavigationCorner this flag can then be accessed by formula:



Macro ShowHideZeroRows

The toggle button should change its - freely selectable and translatable - identifier depending on its status. To enable this, the following settings are still necessary in Customizing:

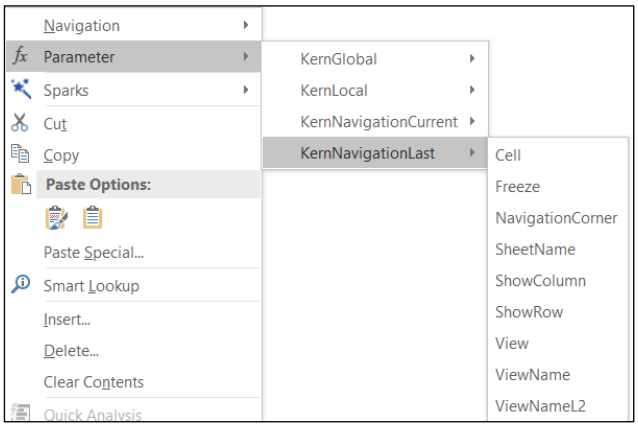
Customizing

Process	SubProcess	Parameter	Key	Description	Value
Common Flags	Common Flags	ID	1		SuppressZeroRows
Common Flags	Common Flags	Flag	1	Toggle-Button	1
Forms and Controls	Forms and Controls	FormName	1		SuppressZeroRows
Forms and Controls	Forms and Controls	ControlName	1		lblShowRows
Forms and Controls	Forms and Controls	ControlCaption	1		Show Null Sums
Forms and Controls	Forms and Controls	FormName	2		SuppressZeroRows
Forms and Controls	Forms and Controls	ControlName	2		lblHideRows
Forms and Controls	Forms and Controls	ControlCaption	2		Hide Null Sums

For the *ControlCaption* parameters, freely selectable identifiers can then be entered in the Value column. If no entries are found here, the default identifiers *Show Zero Rows* and *Hide Zero Rows are* used. The identifier is copied to the {Z_SUPPRESS_ZERO_CAPTION} area depending on the status.

7.6 Context menu

All entries from the ribbon are also available in the context menu (via right mouse button). The context menu contains the same entries and icons as the ribbon and also executes the same macros.



Context menu in the Allevo Master

7.7 Keyboard shortcuts and macros for navigation (Ctrl+Shift+A)

The key combination Ctrl+Shift+A can be used to display the complete structure of the current worksheet. For this, the rows and columns hidden by the navigation are shown again and the set window fixation is cancelled. However, other view properties, such as the column width, are left unchanged.

Alternatively, the *ShowAll* and ShowAll macros are also available.

Note:	Assigning a password for sheet protection in Customizing prevents these functions from being executed.

7.8 Calling SAP functions via the Excel menu ribbon

Normally, Allevo strictly separates functions on the Excel side (e.g. called up via the navigation) from those on the SAP side that are called up via buttons in the SAP toolbar. In this mode of operation, the buttons on the SAP side are always leading.

In individual cases, however, it can make sense to link both worlds, i.e. to trigger functions on the SAP side via an element in the ribbon. Application example: on the Excel side, final calculations are performed that are to be saved immediately in SAP.

Such a coupling can be set up via constant FUNCT_CALL_FROM_EXCEL. In order to use the function, a suitable, customer-specific VBA macro must also exist on the Excel side in which the desired SAP command is integrated (e.g. PLANNING for transferring planning data). This macro can then be assigned to a CustomButton in the "Navigation" worksheet in the "NavigationCustomButton" table.

Custom Button

Custom Button					
SheetName	View	Icon	MacroName	Caption	Tooltip
Allevo		CustomSheetMagicWand	SAPMakro	SAP-Makro	ein SAP-Makro

NavigationCustomButton

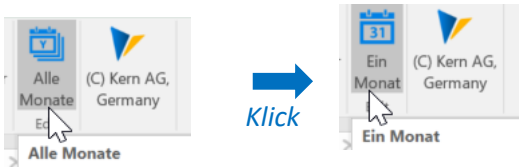
7.9 Saving the navigation settings in SAP

It may be useful to store the contents of the configuration tables *NavigationSheets* and *NavigationViews* in satellite tables on the SAP side and then also load only those selections that are relevant in the respective use case (e.g. depending on object information).

For the comfortable setup of these satellites two groundtables are available: */KERN/U_NAVS01x* for Sheet Settings and */KERN/U_NAVS02x* for View Settings. The last "x" stands for a letter of the alphabet (in the first version of this table it is an "A"). For this, the two tables must be surrounded on the navigation sheet with the corresponding satellite corner.

Standard	ROW						
READ			ZEILE	SHEETNAME	SHEETALIAS	SHEETICON	TOOLTIP
	x		Zeile	SheetName	SheetAlias	SheetIcon	Tooltip
			1				
			2				
			3				

7.10 Clockwise



Clockwise is the name for a button that leads to a different view after each click. All stored views can be clicked through in sequence via the button.

Brief overview

Area	characteristic
Code	
VBA	basKitClockwise
	clsKitClockwise
Settings	
Settings table	ClockwiseSettings
	ClockwiseControl (optional)
	Custom Button

How Clockwise works

When you click on the Clockwise button, a name range in the workbook is addressed via a macro. Each time the button is clicked, the value in the name range is advanced by one value. With the last value, it is normally set back to the first value: 1->2, 2->3, 3->1. A separate name and an icon are set up for each of these values and a view is stored.

Setting up Clockwise

The [CustomButton] and [ClockwiseSettings] tables are required to set up Clockwise. It is also recommended to create an extra table [ClockwiseControl].

1. ClockwiseControl:

Normally, the icon and caption of the custom button must also change when the button is clicked. For this reason, we recommend using a [ClockwiseControl] table (MARE recommendation). The description (FilterCaption) and the icon (FilterIcon) for the respective view are stored in the table. In addition, the values of the views, which are listed in the [ClockwiseSettings] table under Values (see ClockwiseSettings configuration).

ClockwiseControl

Value	FilterCaption	FilterIcon
1	Alle Monate	MultipleYears
2	Ein Monat	Month

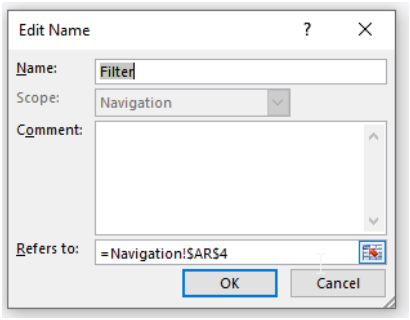
2. ClockwiseSettings:

The structured table with the name [ClockwiseSettings] must be filled on the navigation sheet:

ClockwiseSettings

Name	Values	Case	Range	End	UpdateOff	Sheet
Filter	1|2|3|4	1				
Year	A|B	A				

The "Name" of the clockwise setting refers to a name range. This must always be set up with the scope for a sheet (e.g. navigation).



This name range "counts" in which view the Clockwise is currently set (1->2, 2->3, 3->1). The values can also be alphanumeric, but it is recommended to use numbers.

Configuration

Column	Status	Erläuterung
Name	Obligatory	The name of the clockwise setting. Recommendation: No spaces.
Values	Obligatory	Values separated by "|", which are passed on. The values can be numeric (1|2|3) or alphanumeric (A|B|C).
Case	Recommended	Position within the table in which the namespaces can be stored. Alternatively, the namespace can also be located elsewhere in the master.
Range	Optional	Empty - (default) The name range is identical to the name of the setting. Not empty - The name range differs from the name of the clockwise setting.
End	Optional	Empty - (default) If the name range is set to the last value, the next execution switches to the first value. 1- If the name range is set to the last value, no further switching takes place.
UpdateOff	Optional	Empty - (default) After execution, the currently called view is updated. 1- The view is not updated.
Sheet	Optional	Empty - (default) The name range is located on the "Navigation" sheet (the current sheet in early versions). Not empty - The name range is on a different sheet, e.g. Navigation.

3. Custom Button:

The Clockwise button is set up in the [CustomButton] table.

The MacroName field has to contain the macro "Clockwise" and the name of the ClockwiseSetting (separated by ";").

The icon and caption are each adapted using a formula. The appropriate name can be retrieved from the ClockwiseControl table using INDEX-MATCH, for example, or using a CHOOSE formula:

=CHOOSE(Filter;

"All Values";

"Active Values";

"Key figures")

Custom Button

Custom Button					
SheetName	View	Icon	MacroName	Caption	Tooltip
Allevo	Allevo01	AllRows	Clockwise;Filter	View All	
Details	Details02	Summation	Clockwise;Year	Jahr	

Best Practice for Icon and Caption:

Version 1: only numbers

ClockwiseControl

Row	FilterIcon	FilterCaption	YearIcon	YearCaption
1	AllRows	View All	Summation	Jahr
2	WithoutZerovalues	Only booked	ReportingGraphArrow	Monate
3	Summation	Sum		
4	Edit	Plan		
	AllRows	View All	Summation	Jahr

The value is determined in the totals row:

Numbers (example FilterIcon column): =INDEX([FilterIcon];MAX(1;Filter))

Version 2: Numbers and letters

ClockwiseControl

Row	FilterIcon	FilterCaption	YearIcon	YearCaption
1	AllRows	View All		
2	WithoutZerovalues	Only booked		
3	Summation	Sum		
4	Edit	Plan		
A			Summation	Jahr
B			ReportingGraphArrow	Monate
	AllRows	View All	Summation	Jahr

Letters (example YearIcon column): =INDEX([YearIcon];MATCH(Year;ClockwiseControl[[Row]:[Row]];0))

Reading the values

Formula for reading the value (example FilterIcon column): =ClockwiseControl[[#Totals];[FilterIcon]]

8 Dialog

Dialog03

	READ	WRITE	ROW						
READ				ALLOCATIONSET		OBJECT			ELEMENT
WRITE					ALLOCATIONSET_	OBJECT			ELEMENT
	PROPERTY					DESCRIPTION_OBJECT			
		TIMESET_POST							
	TIMESET								
	DIMENSION	DIMENSION							
	PERIOD	PERIOD							

The Dialog module controls reading and writing in SAP-CO tables.

Brief overview

Area	Feature	Info
Impact Area		
Corner	DialogXX	
KeyPointer	READ	Read from SAP
	WRITE	Writing in SAP
	ROW	From here on the pointers act
Pointer	ACTIVITYTYPE	Activity type (or "_" underscore for the activity-independent part in the case of activity-dependent planning)
	ADPRULE	Rule for performance-based planning 1: 2: 3: 4:
	ALLOCATIONSET	AllocationSet
	ALLOCATIONSETADD	(Shouldn't really be used --- is the old 2nd read which should be done via the AllocationSet itself these days).
	BUKRS	Company code
	COUNTERENTRY	Counter Entry
	CURRENCY	Currency
	DIMENSION	Quantity value unit
	ELEMENT	Element to be planned
	FUNCTIONAREA	Functional area
	OBJECT	Plan object
	OBJECTTYPE	Object type
	PERIOD	Plan period
	PROPERTY	Auxiliary pointer/characteristics for transmitting additional information such as texts
	RECEIVER	Activity Allocation: Receiver Object
	RECEIVERACTIVITY	Activity allocation: Activity type (receiver)

	RECEIVERTYPE	Activity Allocation: Receiver Object Type
	RESOURCE	Resource (internal format)
	SENDER	Activity allocation: Sender object
	SENDERACTIVITY	Activity allocation: Activity type (sender)
	SENDERTYPE	Activity allocation: Sender object type
	TARKZ	Tariff code
	TIMESET	TimeSet
	TRADINGPARTNER	TradingPartner (Profit Center Planning)
	UNIT	Unit of measure
Property: Object	OBJECTTEXT_SHORT	Short text
	OBJECTTEXT_LONG	Long text
	STATUS	Allevo status
	DESCRIPTION_OBJECT	Long text, if available otherwise short text
	CO_DELEGATE	SECPOST object of an object
	CO_DELEGATE_OBJECTTYPE	Object type of an object
	OBJECT_ACTIVE	Characteristic whether the object exists on SAP side 0/1
	CO_STATUS	SAP status
	GROUP_1N	1:N Group to the current object
	GROUP_1N_TEXT	Text for 1:N Group
	OBJECT~<FIELD in the CSKS, PRPS...>	any field from object master data tables
	PROFITCENTER	
Property: Element	ELEMENTTEXT_SHORT	Short text
	ELEMENTTEXT_LONG	Long text
	ELEMENT_TYPE	Element type
	DESCRIPTION_ELEMENT	Long text, if available otherwise short text
	ELEMENT_CATEGORY	Element category
	ELEMENT~<field in CSKA, CSLA,CSKU,SKA1>	any field from element master data tables
Property: Account	SKB1~<FIELD in the SKB1>	each field from accounts master data table SKB1
Property: Activity Allocation	DESCRIPTION_SOBJECT	Sender object text
	DESCRIPTION_SOBJECTTYPE	Sender object type text
	DESCRIPTION_SOBJECLSTAR	Sender object activity type
	DESCRIPTION_SOBJECTOFSET	Offsetting Account Number (offsetting account from COEP)
	DESCRIPTION_ROBJECT	Recipient object text

	DESCRIPTION_ROBJECTTYPE	Receiver object type text
	DESCRIPTION_ROBJECLSTAR	Receiver object activity type
	DESCRIPTION_ROBJECTOFFSET	Offsetting Account Number (offsetting account from COEP)
	RECEIVER_PROFITCCENTER	Recipient profit center
Property: TimeSets	DESCRIPTION_TIMESET	TimeSet description text
Controls		
Macro		
Cado		
UserExits		
Settings		
Configuration sheet	Dialog	
Setting table	DialogAreas	To activate individual corners
Code		
VBA	basDialog	
	clsDialog	

8.1 Functionality

The Dialog module with the associated DialogCorner makes it possible to define posting transactions from the Excel interface directly in and from SAP standard tables (e.g. *COSP*). Different transactions (comparable to SAP transactions) can be triggered simultaneously from one interface.

Note:	In the planning context we call posting to SAP standard tables **standard planning**, it is **configured in** the sheet **|Standard|.** (Planning in customer specific configured tables (satellites) are called **specialized planning**).

Posting parameters are read in Allevo from four hierarchically structured levels.

Higher level parameters overwrite lower level parameters!

For example, a default from the program code is only valid if no other value is defined via TimeSet/AllocationSet/Pointer. Pointers can in turn overwrite the initial object, for example.

1. Predefined in the program code
2. Defined in the addressed TimeSet / AllocationSet
3. Defined via the access to the Allevo transaction (object, version)
4. Defined via pointer in Dialog Corner

Levels 1. to 3. are described in the Allevo-SAP manual. The configuration of the DialogCorer (level 4) is described in the following:

8.2 DialogArea

The configuration of the Dialog module is realized via DialogCorner.

The used DialogCorner are activated in |Dialog| in the [DialogArea] table:

DialogAreas

ID	Active
Dialog01	1

8.1 Table [Dialog Area]

8.3 AllocationSet

An AllocationSet serves as a pointer in the Dialog module. An AllocationSet is created in the layout definition on the SAP page.

AllocationSets, together with TimeSets, define the posting events in the SAP standard tables.

As a good approximation, they correspond to the SAP transactions for posting, for example, the AllocationSet *ACPC* corresponds to transaction *KP06 (1-101) RPK1* in writing in SAP. (Actual Costs Primary Cost)

An AllocationSet defines which table and which operations, load indicators, value type (and record type) are read and written.

Structure of the AllocationSets:

The key of the AllocationSets is made up of 6 digits.

The first two digits specify the AllocationBase, the next two digits specify the AllocationType for **the cost elements**, in all other cases the third and fourth digits remain empty or are filled with " -".

Digits 1-4	AllocationBase	AllocationType
ACPC	G/L accounts / cost elements	Primary costs
ACCC		Secondary costs
ACAI		Power consumption
ACAO		Power outputs
ACOP		Order settlements primary
ACOS		Order settlements secondary
ACCV		Deviations
AT -	Service types	n/a
KF -	Key figures	n/a
TX -	Comments	n/a

The last two digits are free keys that do not necessarily have to be set, they are only used to distinguish custom settings on Allevo SAP side.

Without customer-specific settings for the AllocationType, the following procedure is used:

- **AC:** (Profit center only) For balance sheet accounts the values are read and written as balance sheet data, otherwise as normal values. For cost elements there is no automatism yet (TODO).
- **AT:** Quantities are read (capacities should be read via the dimension C).
- **KF:** Depending on the type of statistical key figure used, either the average (constants) or the sum (total values) is read. If the pointer "REFERENCE_MONTH" is specified, constants can be read for a desired reference month.
- **TX:** n/a

8.4 TimeSets

A TimeSet serves as a pointer in the Dialog module. A TimeSet is created in the Allevo Engine on the SAP side. It defines the **year**, **period** and **version** of the values to be read/written in a column.

The [TimeSets] table is located in the [TimeSets] sheet.

When called from SAP, this is filled with the TimeSet data stored in the layout.

The *HeadlineDescription* column contains a formula for creating TimeSet-dependent headlines. The 6-digit TimeSet designations are then readily placed as area names on the associated *HeadlineDescription field.* These fields then serve as a starting point for designing headlines in |Standard|.

Timesets

Ro	HeadlineDescription	HeadlineRange	HeadlineText	ValueCategor	ValueCategor	VersionFrom	VersionTo	MonthFrom	MonthDesFrc	MonthTo	MonthDesTo	YearFrom	YearTo	TimeSetAdd
1	Ist 1-12 / 2013 V0	A_03	Ist	1		0	0	1	Jan	12	Dez	2013	2013	
2	Ist 1-12 / 2014 V0	A_02	Ist	1		0	0	1	Jan	12	Dez	2014	2014	
3	Ist 1-12 / 2015 V0	A_01	Ist	1		0	0	1	Jan	12	Dez	2015	2015	
4	Ist 1-9 / 2015 V0	A_LTM	Ist	1		0	0	1	Jan	9	Sep	2015	2015	
5	Ist 1-12 / 2016 V0	A_00	Ist	1		0	0	1	Jan	12	Dez	2016	2016	
6	Plan 1-12 / 2015 V0	P_00	Plan	2		0	0	1	Jan	12	Dez	2015	2015	
7	Plan 1-12 / 2016 V0	P_01	Plan	2		0	0	1	Jan	12	Dez	2016	2016	
8	Plan 1-12 / 2015 V2	P_00_2	Plan	2		2	2	1	Jan	12	Dez	2015	2015	
9	Plan 1-12 / 2017 V0	P_02	Plan	2		0	0	1	Jan	12	Dez	2017	2017	
##	Plan 1-12 / 2018 V0	P_03	Plan	2		0	0	1	Jan	12	Dez	2018	2018	

TimeSets

Note:	To simplify the design of the master, this table is often stored in the master already filled with default or test values. These are then overwritten with the data from SAP at runtime.

In principle, the names of the TimeSets are freely definable, but it has proven useful to follow the following naming convention:

TimeSet	Function	
P_00	Plan	Current year
P_01	Plan	Next year
P_02	Plan	The year after next year
P_M1	Plan	Previous year
P_M2	Plan	The year before the previous year
P_YTG	Plan	Current year remaining months (from Perito)

TimeSet	Function	
P_01Q1	Plan	Next year - First quarter
A_00	Actual	Current year to current month (Perito)
A_01	Actual	Previous year
A_02	Actual	The year before the previous year
F_00	Forecast	Forecast current year, A_YTD combined with P_YTG
B_xx	Budget	Analog

8.5 Object and element

In order to uniquely define a posting, (at least) 2 parameters are still missing:

The **object** corresponds to the **place** of posting for example the cost center, the order, the WBS element

The **element** indicates the **type** of posting, typically costs, activity type, key figures

Depending on the posting, additional objects and elements can also be defined, e.g.:

Sender object, sender element, receiver object and receiver element.

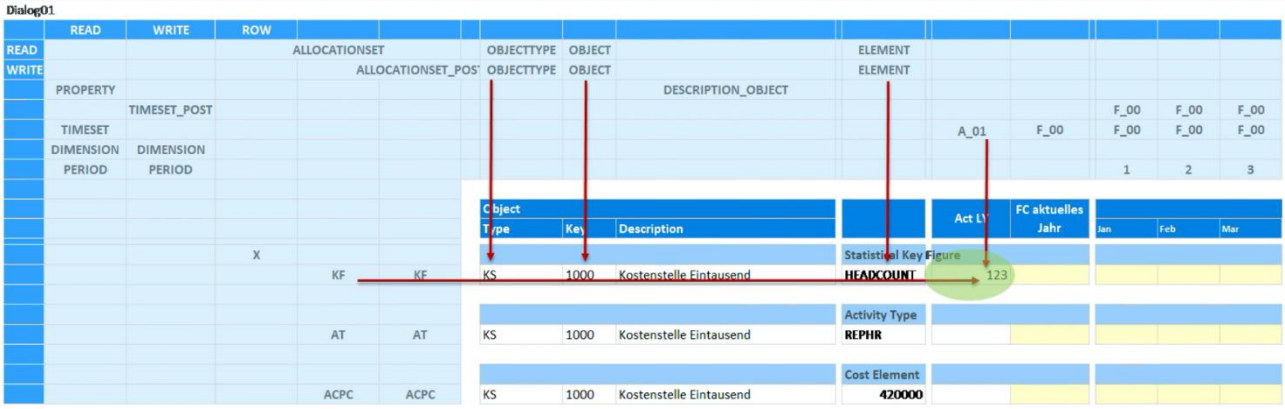
Objects and elements that are transferred via the transaction start are available in the |Info| sheet. Pointers in the dialog corner can overwrite them row- or column-specific.

8.6 The control elements of the DialogCorner

TimeSets and AllocationSets are used as pointers in DialogCorner and thus define the posting event.

Each cell at the intersection of a TimeSet and an AllocationSet can be updated in SAP according to the specifications of these two pointers. In the following example the TimeSet *A_01* (actual previous year) and the AllocationSet *KF* (statistical key figure).

Further pointers specify this posting event, in the example below ELEMENT, OBJECTTYPE and OBJECT define the *HEADCOUNT* key figure and the cost center *1000*.



DialogCorner

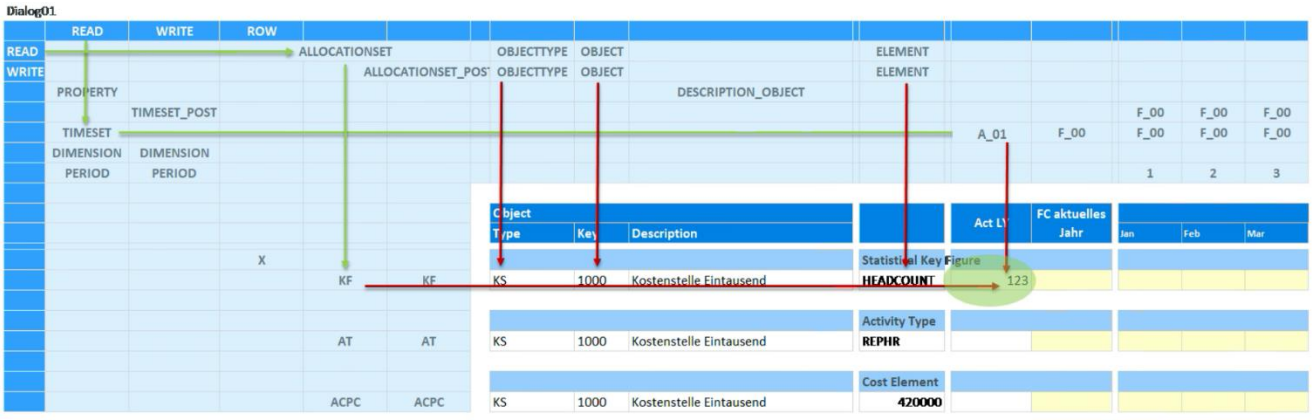
The KeyPointer *READ* and *WRITE* each define a column and row in which the pointers for read and write operations are positioned.

The KeyPointer *ROW* defines the row from which the pointer of the corner is applied.

Writing AllocationSets can receive the suffix *_POST to* instruct Excel to transfer only data that has a specification for this pointer. This is common for *ALLOCATIONSET_POST* and *TIMESET_POST*.

The series READ → TimeSet → A_01 thus defines read access to the TimeSet A_01.

The READ → ALLOCATIONSET → KF series also provides the AllocationSet when reading.



DialogCorner2

8.7 Period

Reading and writing on a monthly basis is defined by the additional pointer *PERIOD*: The right columns in the above example address the months January, February and March. Without *PERIOD,* posting is done on a yearly basis.

8.8 Dimension

The DIMENSION pointer can be used to specify a characteristic for the data type.

Mark-mal	Meaning	Type	L Read P Plan X Both	J Year M Month X Both
V	Value	Value	X	X
V2	Value 2 (for a 2nd currency type)	Value	X	X
VF	Value fixed (activity type dependent planning)	Value	X	X
VV	Value variable (activity type-dependent planning)	Value	X	X
Q	Quantity	Quantity	X	X
QF	Quantity fixed (activity type dependent. planning)	Quantity	X	X
QV	Variable quantity (activity type-dependent planning)	Quantity	X	X
QM	Quantity maximum for statistical key figures (only for years and with new transfer structure with three- row header area in Excel)	Quantity	X	J
PF	Price fixed (tariff)	Price	X	X
PV	Price variable (tariff)	Price	X	X
PU	Price/rate unit (price per xx pieces)	(Text)	L	X
PI	Plan tariff indicator / Price Indicator	Text	X	X
N	Comment	Text	X	J
DKF	SAP Distribution Key (fix) / DistributionKey	Text	X	J
DKV	SAP distribution key (variable)	Text	X	J
C	Capacity (service types) / Capacity	Quantity	L	X
D	Scheduled (activity types) / Activity scheduled	Quantity	L	X
O	Output (activity types) / Output (reserved for future developments)	Quantity	-	-
EN	Equivalence number (service types) / Equivalence (reserved for future developments)	(text)	-	-
FIX	Takes quantity/value from the AllocationSet		X	X
VAR	Takes quantity/value from the AllocationSet		X	X

As can be seen, the first letter in the characteristic always enables a basic, content-based distinction, as e.g. *V* = Value, *Q* = Quantity and *D* = DistributionKey.

Default functions for the characteristic (value key)

It does not always make sense to separate quantities and value columns: especially when planning over periods, this would result in a large number of columns. For this reason, Allevo offers the option of working with default functions for the characteristic.

Example	Without any suffix the SAP side decides on the basis of the respective AllocationSet which content is to be read, e.g. quantity for stat. key figure and activity type or value for primary costs. On the SAP side, the respective characteristic is derived (e.g. *V* for AllocationSet *ACPC* for primary costs).

Allevo uses the pseudo-characteristic *FIX* and *VAR to* distinguish between fixed and variable portions. The last three columns of the following table describe which feature the SAP side uses in this case.

Area	Default AllocationSets	Default (without feature)	VAR	FIX
CO primary costs	ACPC	V value	VV	VF
CO secondary costs	ACSC	Q Quantity	QV	QF
Stat. key figures	KF-R, KF-S	Q Quantity	QV	QF
Service types	AT-M	Q Quantity	QV	QF
Capacities for activity types	ATCAN	C Capacity		
Profit Center (quantities as usual via column key)	ACPCU P (with PC integration)	V value	VV	VF
Profit Center (Obsolete Functions)	B, D, J are only operated out of compatibility with the previous versions and should not be used anymore	Q Quantity	QV	QF

8.9 Activity type-dependent planning (ADP)

Activity type dependent planning is only relevant for cost centers:

Cost elements are not planned general, but divided into three sub-areas:

- **independent** of service type part
- Activity type-**dependent** part **FIX** (per activity type)
- Benefit type-**dependent** part **VARIABLE** (per benefit type)

The allocation to the three shares can either be planned manually or automatically according to a predefined percentage key. The necessary table with the assignment keys is maintained in the Allevo layout in SAP under the item *Rules for Activity Type Depart. Planning.*

Plan performance and tariff

In order to be able to plan a cost recording on an activity type of the cost center, the KL object must exist for the fiscal year and version concerned, i.e. at least one plan price must exist for the activity type or be planned via Allevo.

In Dialog Corner, the AllocationSet *AT is* used to plan the planned output. The dimension *Q* stands for the quantity. Tariffs can be addressed via the dimensions *PV, PF, PU, PI.*

Dialog01

1	READ	WRITE	ROW							
READ				ALLOCATIONSET			ELEMENT			
WRITE					ALLOCATIONSET_POS		ELEMENT			
		TIMESET_POST							PL_01	PL_01
	TIMESET								PL_01	PL_01
	DIMENSION	DIMENSION							PF	Q
			X							
				AT	AT					

1000	Coropate Services	Plan 1-12 2015	
	Pfaehler	Price	New Plan
Activity Type	KP26		
1410	Reparaturstunden	60,0	1.200

Service type and tariff

Note:	Tariffs are only planned if an activity quantity is also planned. If no tariff unit is specified, the value **1 is** set as default.
	Tariffs are planned on an annual basis only.

Set up the activity type dependent planning fields:

By assigning an activity type to an AllocationSet/TimeSet combination via the pointer *ACTIVITYTYPE*, the plan value is planned according to the activity type. The entry of "_" addresses the activity type independent part of the planning.

The FIX and *VAR dimensions* are used to address the corresponding parts.

	READ	WRITE	ROW								
READ				ALLOCATIONSET			ELEMENT		ADPRULE		
WRITE					ALLOCATIONSET_POS		ELEMENT		ADPRULE		
	PROPERTY										
		TIMESET_POST							PL_01	PL_01	PL_01
	TIMESET								PL_01	PL_01	PL_01
	ACTIVITYTYPE	ACTIVITYTYPE							_	1410	1410
	DIMENSION	DIMENSION								FIX	VAR
			X								
				ACPC	ACPC						

1000	Coropate Services	ADP	Act. Independent	1410	
	Pfaehler	Year	Plan	Fix	Var
Cost Element	KP06				
403000	Operating Supplies	1	1.000	100	50

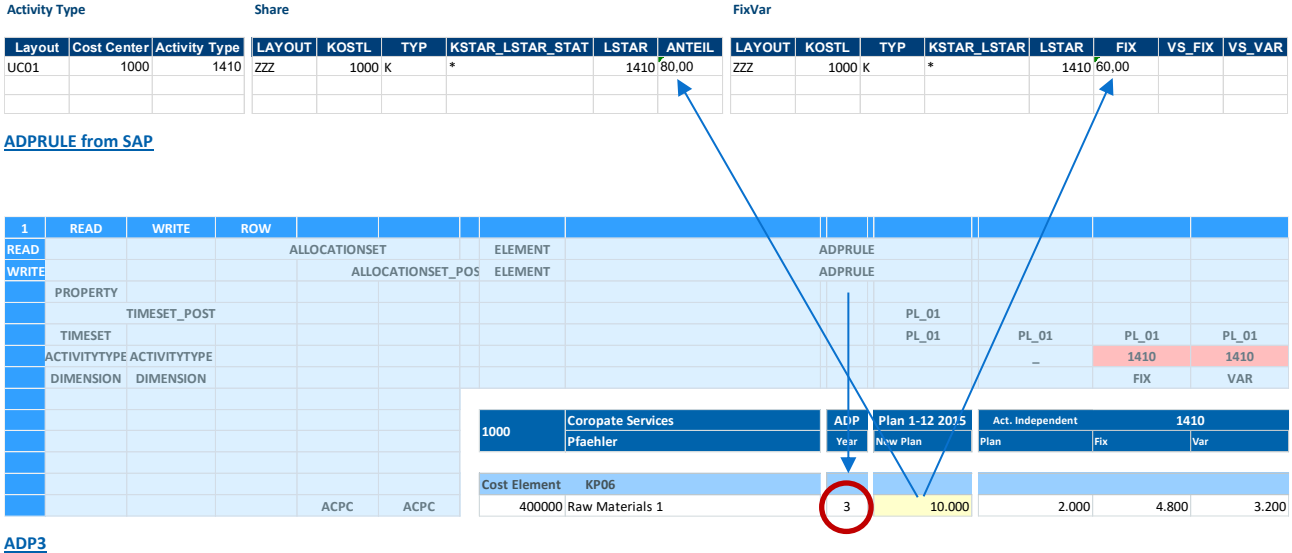
ADP1

Determination of the rules for the LAP per cost element:

The ADPRULE pointer is used to control the LAP process for each cost element. It can either be planned manually in Excel or the percentage distribution stored in Allevo-SAP can be used.

ADPRULE	Rule	Description
blank	No LAP	
1	via Excel; write and read	Activity type-dependent planning via Excel, including readout of the activity type specified on the Excel page
2	via Excel; write	Activity type dependent planning via Excel, only planning (so reading is done activity type independent!)
3	via Rule; write and read	Activity type-dependent planning via rules, including reading of the activity type specified on the Excel page
4	Via Rule; write plan performance and tariff	Performance type-dependent planning via rules, only planning (so reading takes place independent of performance type!)

In the following structure, ADPRULE 3 distribution from the table stored in SAP is used:



Determination of the monthly distribution:

By default, the distribution keys stored in SAP are used. These can also be addressed from Allevo via the dimensions *DKF* and *DKV.* Alternatively, you can also use the Splasher module to distribute the values over the period.

Constants with functions for the distribution key:

PLAN_DISKEY allows you to transfer a distribution key (for fixed and variable) from the Allevo master when using rule planning.

PLAN_DK_FIXVAR herewith the fixed distribution key for cost centers can also be transferred to the variable key.

READ_DISKEY Constant for reading out the last planned distribution keys.

READ_DISKEY_ALL Constant for reading the last scheduled distribution keys.

The following constants control further functions of activity type dependent planning:

BUTTON_LSVAR Button for displaying the rules from the rule set

LSTAR_VARIATOR Activate rules for activity type dependent planning

PLAN_AEQUZIFF Transfer of the equivalence number for activity type from Excel to SAP

TARIF_0_PLAN Service planning for KL objects without tariff

TARIF_KZ Tariff code and tariff unit

READ_ADP Control planning based on reference data

READ_LSTARTABLE Sequence when reading the activity types via satellite.

SAVE_ACTDEP Saving the entries for CY_ADP/CM_ADP

SPLIT_PAROB Activity-based planning with split KL object

USE_DYN_ACTDEP Dyn. range: Display receiver at power consumption

LSTAR_COMMENT_ON Save Allevo comment with sender details (obsolete)

LSTAR_COMMENT_OFF Save Allevo comment without activity type

KL_OBJECT_OFF Ignore KL objects when reading reference data

Activity type-dependent planning over several years: if an activity type is specified via ACTIVITYTYPE, Allevo can also read and plan activity type-dependent without specifying ADPRULE (this should then be done separately for fixed and variable, the total value is always read for months). Optionally, a row can also work with an empty entry in CY_KEYRA in order to read out the activity type-**independent** portions (however, this should then be done using a separate row definition).

8.10 Planning comments

The function of row comments can be activated for each "planning" column definition (DIMENSION "*N*"). This way it is possible to save comments for each fiscal year or for each plan version and of course to read them again, even in case of a multi-year planning.

The row comments are saved for the cost element, activity type or statistical key figure.

When reading out the comments, make sure that the "read" column definition used is set up with a reference to the column definition used to schedule the comments (see Allevo settings on SAP page).

Comment fields can also be used to store information that is not to be understood directly as a planning comment (e.g. to control Excel formulas). In this case in particular, care must be taken to use a suitable Excel cell format (e.g. "Text" instead of "Standard") so that, for example, the content is reproducibly processed in the same way in Excel (equal sign as the first letter would identify the content as a formula or as text content, depending on the case).

The row comments can be integrated into the Allevo report/report interface; they are thus also available for evaluations via common SAP reports (transaction GR55) using document jumps. To set up the interface, see the Allevo SAP manual.

Remove comments

Usually only Excel cells with content are transferred to SAP. This function, which is useful in itself, can lead to existing comments not being deleted later in SAP (so they are there again the next time they are read). This situation can occur if the associated plan column is empty or the comments are entered via a column definition that is only intended for comments.

Solution for this special case: if the *WriteEmptyComments* key is active in the Allevo settings on the customizing sheet, Allevo independently writes the special character '~' in a comment cell as soon as the user deletes the content (thus only applies to columns with suffix "_N"). Since this special character is also passed to SAP, it thus indicates that a comment had previously existed.

Sheet comment (per object)

In addition to the row comment, Allevo allows you to enter a long text once on each Excel sheet (sheet comment). When called up via a single object or via the MultiPage application, this text for the selected object (e.g. cost center, WBS element) is entered and saved. On the relevant Excel sheet, a cell with range name {CC_COMMENT} must exist as cell definition for this (see section 0).

Parent display / editing

Higher-level evaluation and editing of the comments is also possible via Shuttle or the Allevo menu |Satellite tables| (see Allevo SAP manual).

9 Satellite and Suntables

Sat01

	ROW								
WRITE									
READ									

In many cases, satellites serve as a tool to determine the quantity and value components for specific cost elements, activity types, or ratios via detailed plans. Examples:

- Details on cost element planning and activity type-dependent planning
- Leasing and investment planning (incl. derivation of depreciation)
- Maintenance tasks and fleet planning
- Salary costs and employee numbers from a personnel planning
- Resource deployments
- Marketing budgets
- Interface to SAP CO/PA

Note:	In the Allevo environment, the individual use cases of these controlling-specific secondary calculations are referred to as "specialist topic" or "specialist planning".

The interesting thing about these satellites is that the data recorded in these detailed plans can also be stored centrally in SAP and also read back into Excel.

Allevo offers the option of also transferring the data from up to 99 independently definable Excel tables to SAP, where it can be stored centrally in corresponding SAP tables. The Allevo SAP manual provides detailed information on this.

Structure of a satellite table

In SAP, the data is in the same table across object types, objects, versions, and fiscal years.

Each satellite table has the following field structure:

Area	Feature	Info
Code		
VBA		
Settings		
Configuration sheet		
Setting tables		
Controls		
Macros		

Impact Area		
Corner	SatXX	
KeyPointer: Row	READ	Read from SAP
	WRITE	Writing to SAP
	FORMULA	(optional)
KeyPointer: Column	MANDT	Client
	KOKRS	Controlling area
	SETCLASS	Setclass
	COOBJECT	Object
	PJAHR	Business year
	VERSION	Plan version
	ZEILE	Row
	Custom fields	In any number

Note:	Satellites must be activated in the respective constant for the planning layout so that Allevo takes them into account when reading and writing (constant ACTIVE_SAT).

The data of a SAP satellite table are mapped in Excel via two specific areas each (e.g. for satellite 1 via SAT01Row and SAT01Column), the row and column axis of the respective SatelliteCorner. The column axis controls, among other things, the field mapping, i.e. the assignment of the Excel columns to the fields of the SAP table.

As a rule, the fields from field no. 7 (ZEILE) onwards are transferred to Excel, the selection of the data records is made depending on the fields 1-6 on the initial settings of the Allevo layout see sheet |Infos|)

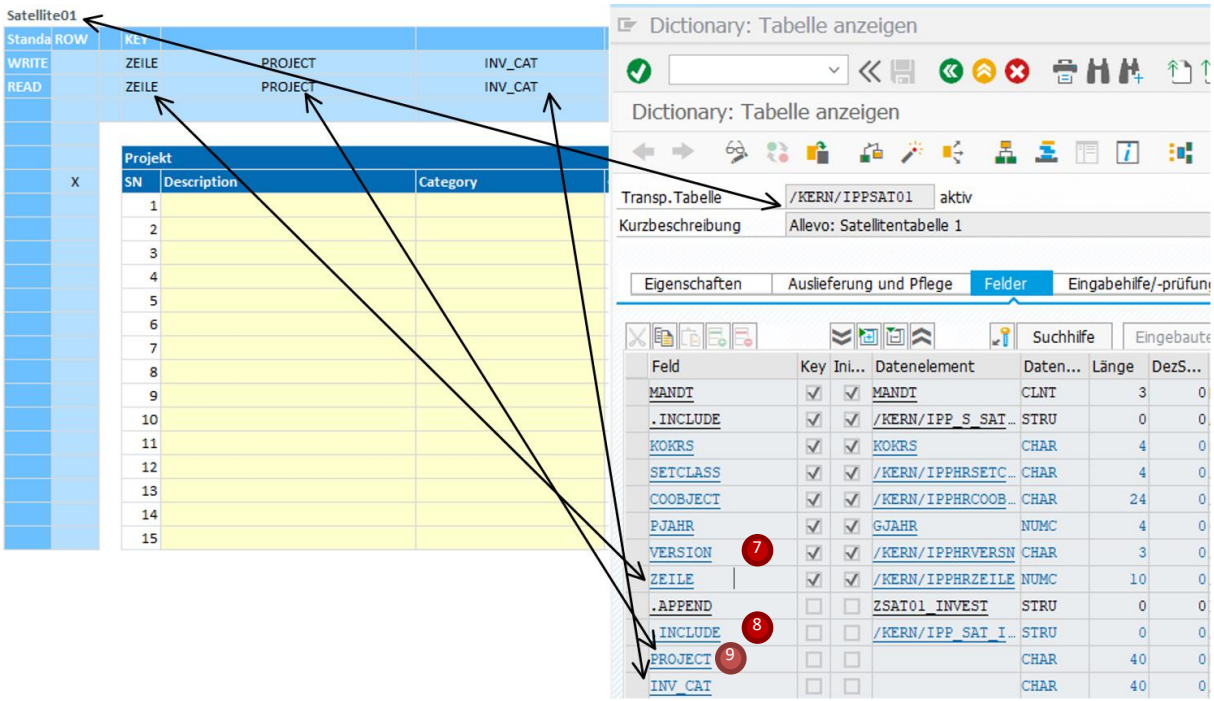
The SatelliteCorners in the Allevo Master are used to selectively transfer data to/from an SAP satellite table.

The KeyPointers *READ* and *WRITE* in the **column axis →** control the assignment of the columns to be transferred from/to SAP based on the key names of the individual satellite fields. The KeyPointer *ZEILE* has a special meaning. The satellite areas on Excel side often start with this field; however, when working with more than one object, the other key fields of the satellite table (field numbers 1-6) should also be considered.

The control element for the **row axis** ↓ marks the data area of the satellite and thus the cells in which data is exchanged with the satellite table. The KeyPointer *ROW* marks the row below which the data area begins. The range usually ends with the lower end of the row axis (exception in the modes *All* and *Lean*, in which the data range is also searched in further rows).

The following figure explains the connection between satellite table in the SAP system and the display via the Excel ranges. The axes formatted in blue are the Excel name ranges (vertical: {SATxxRow} ↓, horizontal: {SATxxColumn} →). The functional mapping of the respective columns and rows in these ranges by KeyPointers are highlighted in the following graphic. The top arrow represents the connection between satellite number in Excel and the table key in SAP. The other arrows form a reference to the corresponding columns in

Excel. When reading data from SAP, the number is usually set according to column 7 (= field ZEILE) of the satellite table. However, using special functions, it is also possible to insert the number to the exact row in Excel (see notes below). It is also possible to set the pointer ZEILE for READ/WRITE in different columns.



Satellite tables, mapping on Excel and SAP side

9.1 Read and write satellite areas (sequential)

In most applications, data from a satellite table is inserted sequentially into the associated satellite area (or entered there). The sequence when inserting the rows is defined via column 7 (= row), because data is only inserted into those Excel rows with an entry in *READ/ZEILE.*

In more complex applications of Allevo with different objects or object types on a sheet, the data of the satellites for different characteristics must also be read or written (e.g. controlled via the Allevo constant GRP_READ_SATxx). In this case, the row number is no longer unique and additional columns from the index of the satellite table must be displayed in the Excel satellite area. *SETCLASS* and *COOBJECT* are mandatory.

Note:	In principle, all fields of a satellite can be displayed in Excel. However, details such as object, year and version are constant for the entire planning in standard Allevo applications anyway and are automatically filled when saved in SAP (applies in particular in MultiPage mode with one object per sheet). In MultiObject mode, the index fields 1 to 6 should always be kept on the Excel side as well; especially if cross-object data are processed (see F1 help for the constants SATxxSELECT or GRP_READ_SAT).

Saving satellite data

When saving to SAP, only the rows with an entry in WRITE/ZEILE are transferred to SAP and saved in the relevant satellite table. On the SAP side, all satellite data for the current object (as well as year, version, etc.) are deleted: at the end, therefore, the satellite table contains an exact image of the data that was previously displayed in Excel. In MultiObject mode, satellite data for different objects/object types are also stored as required (the associated fields must therefore also be filled on the Excel side).

9.2 Use of "Structured Tables" for satellites

A structured table is defined with the Excel key combination Ctrl+T. It enlarges automatically as soon as values are added directly below, to the right or to the left.

With limitations, Allevo also supports these functions for satellite areas; in particular, formulas can have an impact on performance (for large data volumes).

Note:	In "Structured Tables", formulas of a cell are automatically applied to all cells of the column. There are cases where this function called "AutoFill" is not desired (e.g. because of performance when reading many satellite rows). In Excel, the setting is made for the respective workstation (see setting "Fill formulas in tables" in the Excel options under "Document check and auto correction"). It makes sense to reset the parameter to the original entry when you exit Allevo. That is why there are two Allevo macros: • *DeactivateAutoFillFormulas* deactivates the auto-filling of formulas; the macro should be entered at the WB_OBEN or BEF_READ events. • *ResetAutoFillFormulas* resets the Excel setting to the previous state at the workstation (so the macro should be entered at WB_CLOSE).

9.3 Execution variants of the write and read functions

Overview

By default, satellite data is inserted sequentially into the associated Excel range. In standard mode, only Excel rows for which a value is entered in column "ZEILE" are considered (it does not matter whether this value is numeric or alphanumeric). When saving, Excel transfers the data to SAP in the order in which they appear in Excel. In this sense, the data transfer is sequential when reading and writing.

Satellite data does not have to be listed strictly sequentially, but can also be divided into different blocks within the read/write areas (as long as a continuous column division is given). Then, for example, headings or other rows can be located in the satellite area, which are not overwritten when reading reference data and are also not transferred to SAP when transferring plan data. The following figure shows the basic structure, whereby the field names from the satellite could also be used instead of the indices in the header area.

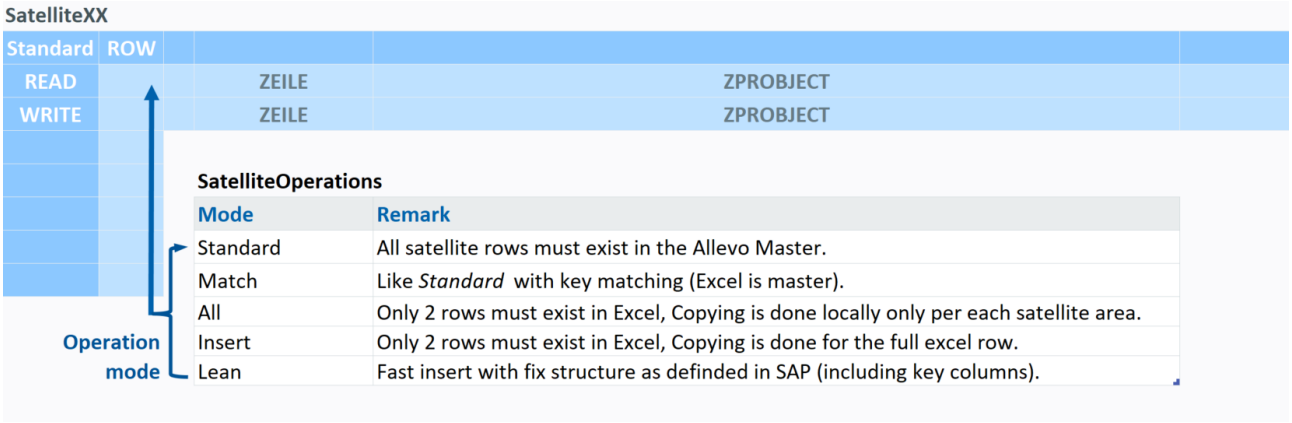
Sat02

Standard	ROW							
WRITE			ZEILE	PERSNR	NAME	ENTRY	EXIT	FTE
READ			ZEILE	PERSNR	NAME	ENTRY	EXIT	FTE
	X							
			Angestellte					
			1	4.711	Miller	01.01.2021		100,00%
			2	5.345	Mayer	01.10.1998		50,00%
			3	2.764	Schmitd	01.01.1950	31.12.2015	100,00%
			Auszubildende					
			4	6.528	Schreiber	01.02.2022		50,00%
			5	5.423	Liebke	01.01.2020		50,00%
			6					
			Praktikanten					
			7	9.987	Hamann	01.07.2011		100,00%
			8					
			9					

Writing/reading area with divided contents

The data transfer between Excel and SAP can be done in different ways and is defined at the top left intersection per satellite. Different variants are available for this purpose.

Here is an excerpt from the Allevo Master:



SatelliteXX

Standard	ROW		
READ		ZEILE	ZPROBJECT
WRITE		ZEILE	ZPROBJECT

SatelliteOperations

Mode	Remark
Standard	All satellite rows must exist in the Allevo Master.
Match	Like *Standard* with key matching (Excel is master).
All	Only 2 rows must exist in Excel, Copying is done locally only per each satellite area.
Insert	Only 2 rows must exist in Excel, Copying is done for the full excel row.
Lean	Fast insert with fix structure as definded in SAP (including key columns).

Excerpt from the "Satellite" definition sheet in the Allevo Master

The most important features of the respective variants are listed below:

“Standard" mode

The Excel area must be set up in advance for the expected number of rows.

Basic functions:

Rows are inserted in the order they come from SAP.

The relevant columns are taken from the header information (separated between read and write, if necessary).

When writing satellite data to SAP, the row numbers are recounted before being stored in the SAP database table.

Other properties in standard mode:

Column-by-column transfer for good performance

Subheadings are possible by empty field in the column with pointer ZEILE

“All” mode (formlery InsertNewSatRows)

When reading, all rows supplied by SAP are entered, regardless of the dimension of the "SatelliteCorner". No further contents may be below the satellite: they would be overwritten.

The rows are inserted after reading from SAP without regard to existing formatting; for the same reason, subheadings are excluded.

All overwrites the entries of column 7 (= field ZEILE) and renumbers the row specification when saving.

It is recommended to write the data in a structured table, as this simplifies the later access via Excel functions. The table is extended to the number of rows transferred from SAP.

Note:	When using the FormulaCorner with satellites in the "All" mode, the KeyPointer *RowAll* must be set instead of *Row*

"Insert" mode (formerly InsertNewExcelRows)

Basic function as for *Standard*: but here Excel rows are copied completely, corner areas can thus grow along with them

No row numbers are inserted, but all entries are copied from row 1. If necessary, an entry is also copied into column *ZEILE,* e.g. to enable automatic row numbering.

The function for copying whole rows is activated via the entry *Insert* in the top left cell of the intersection of the two satellite axes.

"Lean" mode

This mode is particularly performant and also the means of choice in connection with PowerPivot. All data is inserted into the Excel area as it is provided by SAP; the rows are therefore not renumbered (in column 7).

On the Excel side, there is no assignment of columns via header specifications: consequently, the column structure in Excel is exactly the same as on the SAP side (including all key fields 1 to 7).

No other content may be placed below the satellite; it would be overwritten if necessary.

Within the SatelliteCorner a structured table with at least one data row must be created: The number of columns must be the same as in the satellite table on the SAP side. No corner areas will be extended in the process.

The function for easy and fast reading and writing is activated via the *Lean* entry in the top left cell of the intersection of the two satellite axes.

Mass data for reporting

If the data of a satellite should only be evaluated in PowerPivot (without clipboard on an Excel sheet) the constant SATxx_FILE is available alternatively.

In this case, the data is transferred via a temporary CSV file (see F1 doc for constant). Advantages especially with mass data: again higher performance and a much smaller Excel file.

"Match" mode (formerly MatchExistingRow)

This mode allows a row-exact reading/writing: the row key in column "ZEILE" has a central control function here. According to the row number, which is predefined in Excel, the satellite rows are entered.

The link to the matching Excel row is made via the specified Excel row number in column *ZEILE* and the corresponding number in the SAP satellite table, i.e. with the number stored in the field *ZEILE* of the satellite table.

This exact assignment then also allows to arrange satellite rows in predefined order on the Excel sheet (e.g. 1,5,2,23, ...).

Example:	The row number can also be derived from other parameters on the Excel side, such as via a formula with reference to the cost element, in order to place a satellite data row exactly over the Excel row with matching cost element (in current Allevo versions, a maximum of 10 digits is possible).

The function for row-exact reading and writing is activated via the entry *Match* in the top left cell of the intersection of the two satellite axes.

Note:	The function for row-exact reading and writing also requires activation for the respective satellite on the SAP side (see constant SATxxSELECT).

	Background: in the standard case, Allevo would renumber all rows when reading and writing. This is not the case here, of course. In an extended variant, the constant mentioned above not only allows the exact specification of the relevant rows for selection; alternatively, Excel can also specify other selection characteristics in the satellite key in this way (similar to Allevo Dynamic functions).

Operation mode depending on READ & WRITE

If - as in the previous versions - the mode is entered in the cross box, it applies to both reading and writing to SAP. From version 4.3, it is also possible to specify different modes for reading and writing.

The desired modes are entered under Pointer MODE:

SatelliteXX

Standard	ROW	MODE		
READ		Standard	ZEILE	
WRITE		All	ZEILE	
	X		01	Invest
			Row	Description
			1	
			2	
			3	

Not all combinations make sense:

- The Insert mode, for example, does not make sense as a WRITE mode, as it only differs from the standard in that the rows are inserted when reading to Excel.
- In Lean mode, columns/rows are not assigned via the pointers, which is why it does not make sense as a WRITE mode in combination with another READ mode.
- Match mode is also not suitable for combinations

The following combination options are available:

Translated with www.DeepL.com/Translator (free version)

READ	Standard/Insert	All	Lean	Lean
WRITE	All	Standard*	All	Standard*

* Der READ-Modus hat nur ein "Rumpf-Corner", der WRITE_modus braucht den ganzen Corner, deshalb würde das nur zusammen mit einem Stretchcorner funktionieren.

Note:	For deleting and resetting a satellite range: When reading a satellite, any content on the Excel page is deleted. Which columns/rows are deleted depends on the satellite mode. As of version 4.3, the READ mode is now always decisive. The same applies when resetting the satellite in the panel.

Row-exact reading for multiple objects simultaneously

When working with different objects or object types on one sheet, the data of the satellites for these different characteristics must also be read or written (e.g. controlled via constant GRP_READ_SAT on SAP side, see F1 documentation for this).

In this case, the *ZEILE* column is no longer sufficient for exact positioning; instead, specifications in *COOBJECT* (and also in *SETCLASS* for different object types) must be included in the positioning. For this purpose, these additional specifications can be configured via the KeyPointer *SORTING.* Here the basic structure:

Satellite01

Matchi	ROW			
WRITE			ZEILE	COOBJECT
READ			ZEILE	COOBJECT
SORTING			1	2
	X			

Projekt	
SN	COOBJECT
1	0000001000
1	0000001200

Line-by-line reading over multiple objects

On this basis, Allevo tries to find the matching row on Excel page when reading each row of a satellite. The following finding rule / sequence applies:

First mapping field stored in SORTING (e.g. ZEILE in the example above)

Second mapping field stored at SORTING (e.g. here COOBJECT)

Important: a field to be used for SORTING must also be read from SAP (entry at READ must exist).

9.4 Use case: Number of rows in the data area

In the execution mode "Standard" the number of rows in the data area of Excel is designed according to the expected data volume. This can be difficult in individual use cases (e.g. dynamic selection of data via MOD or ProCED).

Then the following Allevo functions can be helpful:

Switch to All or Insert mode: to expand the row range dynamically, i.e. depending on the number of rows supplied by SAP.

Alternatively, Allevo offers a control function that is run through after each data transfer of satellite data to Excel: if not enough rows are available in the Excel satellite area, an info is returned to SAP and an error message is issued there (as is usual with SAP in the status bar at the bottom left).

9.5 Suntables

Suntables are a simplified form of satellite tables for the storage of comprehensive settings, without reference to object, version, year etc.. They therefore contain the client (MANDT) as the only key field. As with the satellites, there is a customer-specific table for each Suntable on the SAP side.

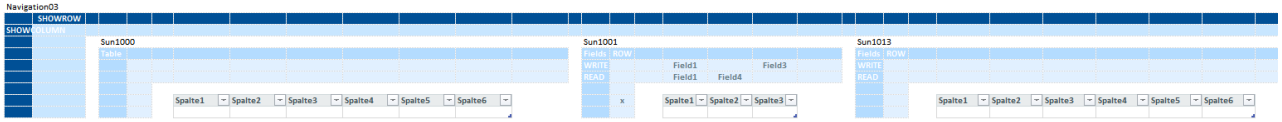
Sun1000

	ROW			
WRITE				
READ			LOCATION	TIMEZONE
	X			
			East	GMT+8
			West	GMT-3
			North	GMT-2
			South	GMT+2

Suntable

By and large, SUN Corners are set up like satellites. However, there are the following differences:

- SUN Corners are not created in cascade, but in series. The SUN corners must therefore be on a separate sheet.
- The numbering of the SUN tables in SAP starts at 1000, which means that the corner areas must be numbered accordingly: SUN1000Column or SUN1000Row.
- The SUN-Corner knows only two modes: Table and Fields.
- In principle, structured tables are expected.



The SUN modes

Table

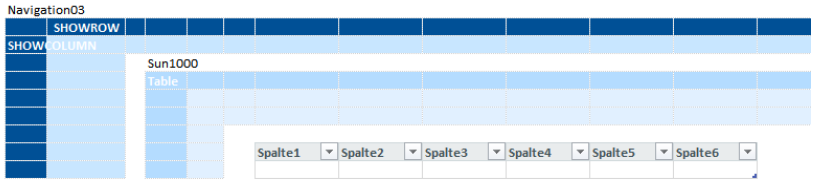
"Table" corresponds to the "Lean" mode for satellites.

The data of the complete SAP table is inserted into a structured table.

The number of columns must be specified. The number of rows is determined by VBA and the table is extended accordingly.

Use case is e.g. PowerPivot.

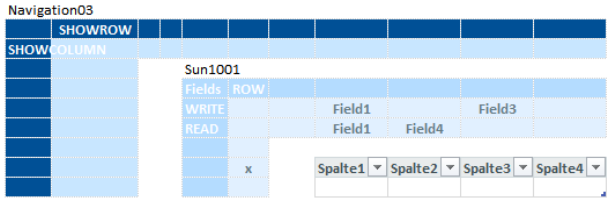
The table should have only one row. The column structure of the table must be exactly the same as on the SAP side. The corner itself can be kept small, because neither its row nor its column dimension is considered by VBA. No pointers are necessary, they are all ignored by VBA.



Fields

"Fields" essentially corresponds to the "All" mode: the row dimension of the corner is ignored by VBA, i.e. the full number of rows from SAP (when reading) or from Excel (when planning) is always processed. When creating the corner, it is sufficient to provide one row for the data area.

The biggest difference to the satellite with mode "All" is that there is no predefined key field "Row". VBA therefore does not generate key values for this or any other field even when saving to SAP.



10 Dynamic

Dynamic01

	DYNAMIC	ROW							
GETKEYS									
FILLKEYS									

By means of the Dynamic Module, the row and column structure dynamically adapts to the object / element structure read from SAP. The positioning of the areas is controlled via pointers in the corner. A corresponding StructureCorner then takes over the formatting and conversion to formulas of the summary rows.

The dynamic rows are generated in each case with reference to the Allevo initial object; for example, the cost center. If a 1: n group is stored there, the dynamic rows are determined aggregated for all objects of the associated group: i.e. separately according to sender and receiver activity types, but not separately according to object.

In MultiPage mode, the rows of dynamics are created with reference to the object per sheet.

Brief overview

Area	Feature	Info
Code		
VBA	clsDynamic	
	clsDynamicRanges	
Settings		
Configuration sheet	Dynamic	
Setting tables	DynamicAreas	
Controls		
Macro		
Cado	CadoAfterTransferDynamic-ToExcel	
UserExits	AFT_DYNAMIC	
Impact Area		
Corner	DynamicXX	XX = 01, 02, ...
KeyPointer	GETKEYS	Controls the transfer of structure data to Excel via the specification of pointers, which determine the positioning in columns. Alternatively, GETKEY can also be used for dynamic column layout.

	FILLKEYS (optional)	Entries for the KeyPointer FILLKEYS instruct Dynamic not to build the whole structure, but to restrict it to selected elements in the columns with the pointers belonging to FILLKEYS.
	DYNAMIC	The dynamic areas are mapped by so-called "DynamicSets", which are arranged in the Dynamic Corner under the "DYNAMIC" KeyPointer. A DynamicSet is defined in Allevo SAP.
	ROW	An X marks the first row from which the Dynamic is executed
Pointer: GETKEYS	ALLOCATIONSET	These pointers corresponds to the pointers with the same name in DialogCorner
	ALLOCATIONSET_POST	
	ELEMENT	Element or summation level
	ELEMENT_ID	Element (all hierarchy levels)
	ELEMENT TEXT	Element text
	OBJECT	Object or summation level
	OBJECT_ID	Object (all hierarchy levels)
	OBJECTTYPE	Object type
	OBJECTTEXT	Object text
	PROFITCENTER	Reads associated profit center for cost elements and statistical key figures and object hierarchy
	RECEIVER	These pointers corresponds to the pointers with the same name in DialogCorner
	RECEIVERACTIVITY	
	RECEIVERTYPE	
	SENDER	
	SENDERACTIVITY	
	SENDERTYPE	
	TIMESET	Corresponds to DialogCorner -- when swapping the "Normal" axis
	CATEGORY	Element type of the respective element: Cost element type from SAP, Statistical key figures: A=Average S=Sum
	SUMLEVEL	Level of the respective summation, can be used directly by the StructureBuilder
	TOP_KSTAR_GRP	Old MOD field IS_TOP_KSTAR_GRP
	UPPER_HIERARCHY	Parent group
	UPPER_HIERARCHY_TEXT	Parent group text
	SAP_ELEMENT	SAP element from an Allevo element
	SAP_KATYP	CATEGROY of the SAP element from an Allevo element

	SAP_AS_READ	ALLOCATIONSET for an SAP element from an Allevo element
	SAP_AS_POST	ALLOCATIONSET_POST to an SAP element from an Allevo element,
	SAP_ELEMENTTEXT	Text of the SAP element from an Allevo element
Pointer: FILLKEYS	The following pointers overlay the object selection from the StartScreen (for MultiPage & MOWO)	
	OBJECT	Object
	OBJECTTYPE	Object type
	The following pointers correspond to the filter settings of the DynamicSets in Allevo-SAP:	
	ALLOCATIONSET_POST	AS Plan
	ALLOCATIONSET	AS Read
	FILTERTYPE	Object type
	FILTER	Filter in a Dynamic Set
	HIERARCHY	Sort
	ELEMENTSORT	Element sort
	PRESELECT	Preselection
	SUMCHAR	SumSign
	RESOLVELEVEL	GRP resolution level
	RESOLVETYPE	Type of resolution
	CHECKTS	TimeSets for validation
	SATNR	Satellite
	AGGREGATION	Compaction in dynamic range
	TIMESETS	TimeSets for Dynamic Ranges
Pointer DYNAMIC (Every dynamic set created in Allevo-SAP can be addressed here - only the most common ones are listed here)	AO	Activity Output (credit from activity allocation to cost centers). Row definition K allows to read values for power output, from version 3.5 also planning is possible (see notes below).
	AI	Activity input (Source object / via Settlement) Power consumption / load from secondary job accounting (As a result of a collective settlement over several objects, there may also be entries with an empty source object; displayed as a separate row in the Allevo dynamic).
	AS	Activity Input (Partner Object / from Direct Allocation) Activity Input / Load from Activity Allocation
	CH	Cost Element Hierarchy
	CE	Cost Elements
	KF	Stastical Key Figures

	KH	Key Figure Hierarchy
	OH	Object Hierarchy
	BS	Balance Sheet Structure

10.1 Controls

The dynamic controlled page layout is done automatically when the master is loaded and cannot be triggered further.

10.2 Settings

KeyPointer

The field structure is controlled via the structures found in SAP using the FILLKEYS and DYNAMIC KeyPointers. GETKEYS determines which structural elements are used to fill the columns, while DYNAMIC controls the row structure analogously to a configuration stored in Allevo-SAP (Dynamic Sets).

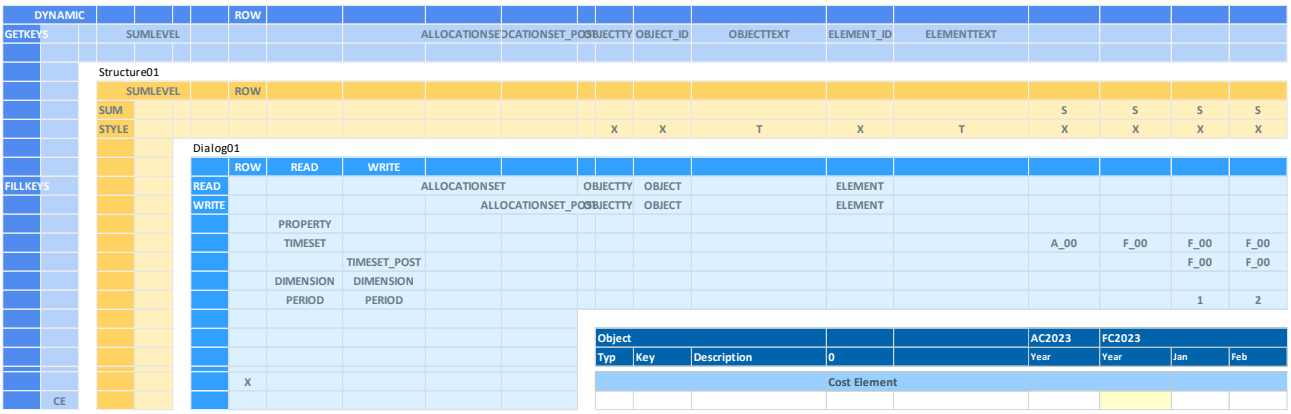
The pointers under FILLKEYS allow overwriting the settings in the respective DynamicSet.

Dynamic makes the following PROPERTIES in the Dialog Corner redundant, these are to be removed in the dialog for performance reasons:

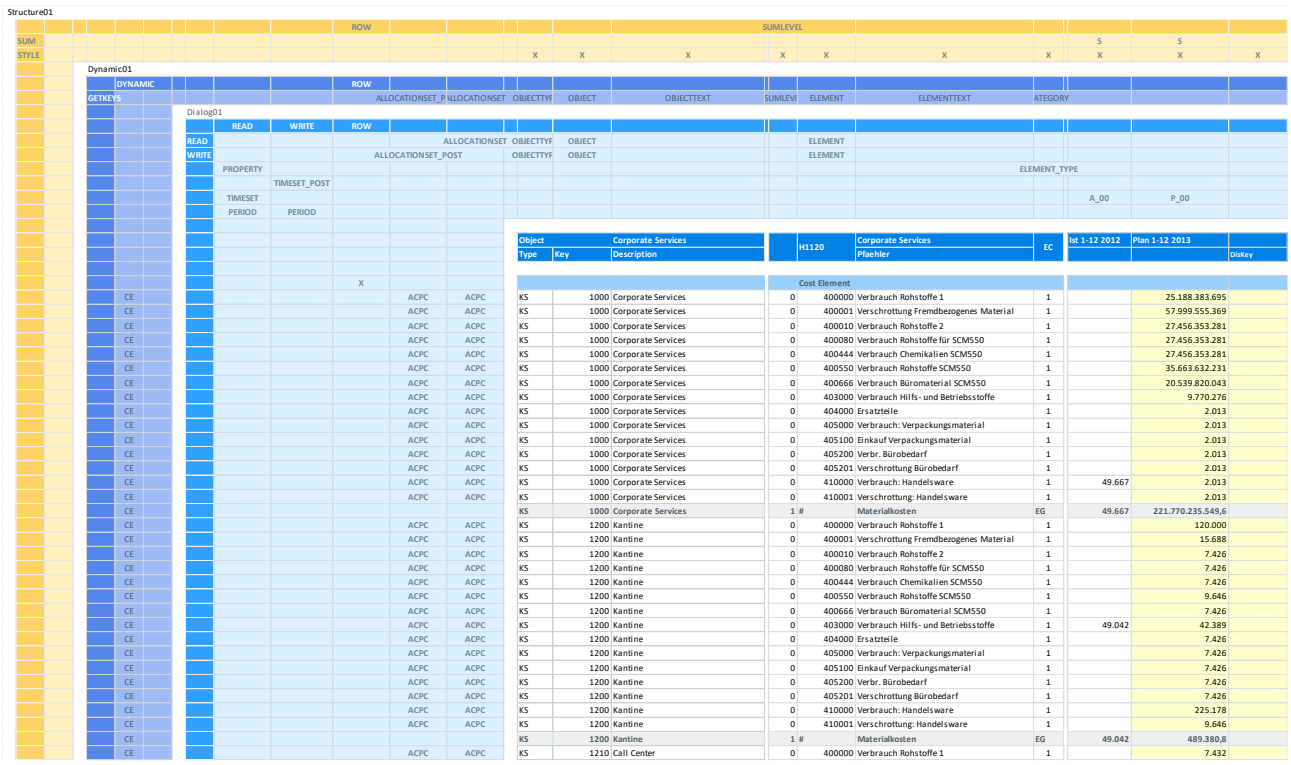
Property in dialog	Pointer in Dynamic
DESCRIPTION_ELEMENT	ELEMENT TEXT
DESCRIPTION_OBJECT	OBJECTTEXT
DESCRIPTION_OBJECT	CATEGORY

DynamicCorner

The following figure shows the corresponding section of the Allevo master. The row structure is dynamic when called from SAP:



Dynamic areas in the Allevo Master



Dynamic areas in the Allevo Workplace

11 Style

Style02

	STYLE								
STYLE			3	3	2	1	4		
	2								
	2								
	1								
	1								

Brief overview

Area	Feature	Info
Code		
VBA	basKitStyle	
	clsKitStyle	
Settings		
Configuration sheet	Style	
Setting tables	StyleAreas	
	StyleSetDefinitions	
Controls		
Macro	StyleStart	
Cado	CadoBeforeStyle	
UserExits	Aft_Style	
Impact Area		
Corner	StyleXX	
KeyPointer	STYLE	

11.1 Structure of a StyleCorner

Functionality

The StyleCorner can be used to dynamically customize cell style sheets and cell protection.

The formats are determined via the configuration in the corner. Here, configuration pairs are read from column and row and the formats are set by means of intersection formation.

Set up corner

The corner is set up as usual: {StyleXyColumn} and {StyleXyRow}. The desired StyleCorner are activated via the table [StyleAreas]:

StyleAreas

Id	Active
Style01	1

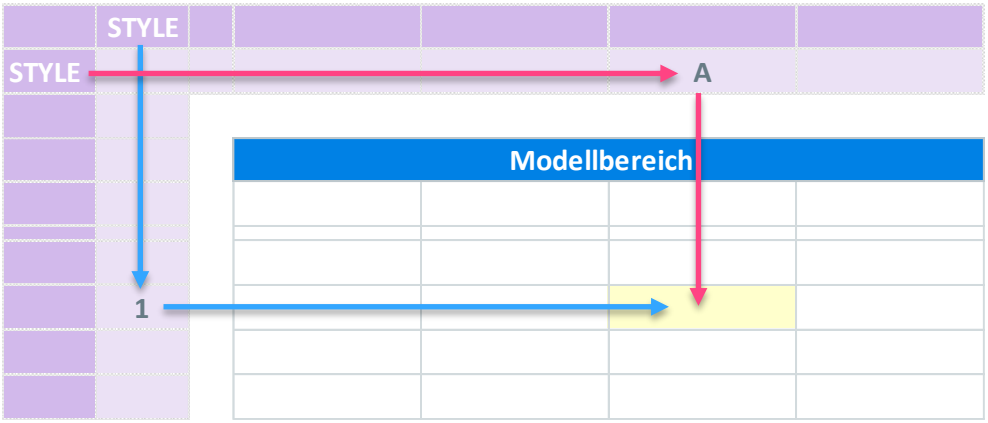
StyleAreas table

With the macro *StyleStart* all StyleCorner of the table are executed. The macro must be entered at a UseExit.

In both column and row alignments, the configuration keys can be specified.

Example:

Style01



Corner Style01

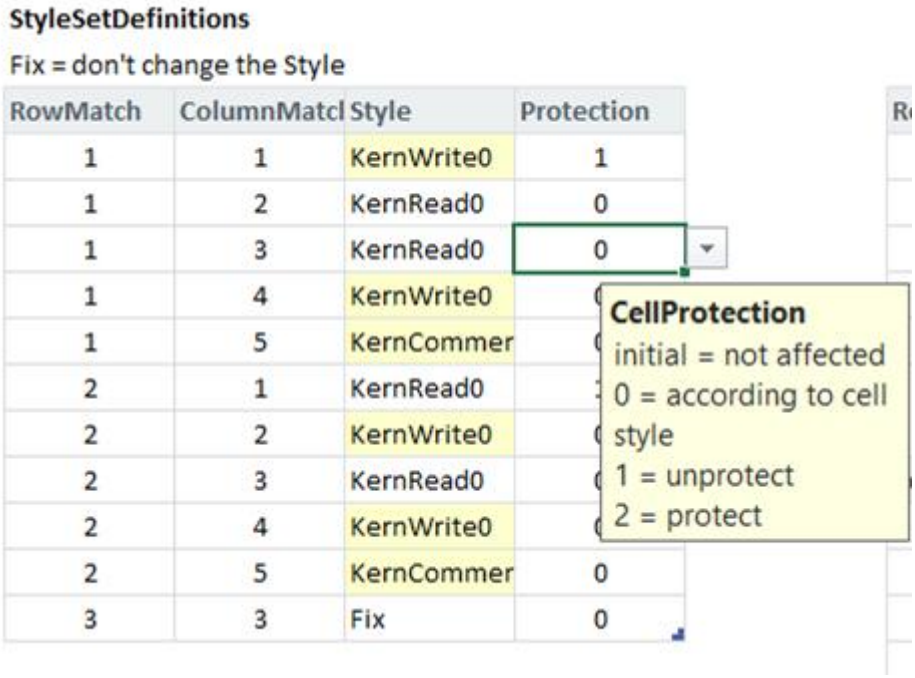
The configuration key combinations (per column and row) must be defined in the [StyleSetDefinitions] table. The configurations defined in this table apply to all corners defined and active in the [StyleAreas] table.

11.2 KeyPointer of the Style Corner

STYLE

Columns or rows can be marked here to give them a certain cell "style".

The configuration key combinations (per column and row) must be defined in the [StyleSetDefinitions] table. The configurations defined in this table apply to all corners defined and active in the [StyleAreas] table.



StyleSetDefinitions table

The configuration of the assignment key per row and column is done via the specifications: *RowMatch* and *ColumnMatch*. The assignment keys can be any (letters or numbers greater than 0).

Note on the assignment key	From version 4.3.2 an assignment key with the value “0” is interpreted as an empty field and therefore ignored.

The *Style* column specifies the Excel cell style to be used (can be replaced by your own as desired).

Only the style name of the cell is read and not the cell format.

The entry *Fix* in *Style* allows to keep the current cell formatting (no change by the corner).

A cell protection rule can be defined in the *Protection* column:

1= Unlock cell

2= Lock cell

0= Apply cell protection setting of the style

11.3 Extension of the configuration table [StyleSetDefinitions] by the optional column Area

By default, all pointer combinations defined in the [StyleSetDefinitions] table are valid for all corners defined and active in the [StyleAreas] table.

If the configuration table [StyleSetDefinitions] is extended by the optional column *Area*, individual style corners can be entered and addressed:

StyleAreas

Id	Active
Style01	0
Style02	1

StyleSetDefinitions
Fix = don't change the Style

Area	RowMatch	ColumnMatch	Style	Protection
*	1	1	KernRead0	1
Style01	1	2	KernWrite0	0
Style02	2	2	KernHeadline1	

StyleAreas and StyleSetDefinitions table

Note:	If the column is not present, the configurations are valid for all active StyleCorner. If the column is present, the "*" entry applies to all StyleCorner. If nothing is specified (neither "*", nor StyleCorner name), the configuration is ignored.

11.4 Only consider active lines (from 4.3)

To insert styles only in the active row(s) of the active sheet, the StyleActivateInline function can be used (e.g. in combination with the PickList or the Jumper).

It is integrated via UserExit:

Customizing

Process	SubProcess	Parameter	Key	Descriptio	Value
UserExit	Macro	AFT_JUMPER			StretchStart
UserExit	Macro	AFT_STRETCH			StyleActivateInLine Style01

11.5 Set the entire style to read-only (from 4.3)

When using the Panel or the One-Page-Multi, no plan data can be transferred as soon as the user is in a group. The StyleSwitch macro does not change the cell formatting of the individual fields, but sets the entire form template to read-only mode. This is done by

- a visual change, i.e. the background color is adjusted
- a technical lock, i.e. the style is given the read-only feature

4100	Techn.Service - 1	EC	Ist 2012	Plan 2012	IST 1-003 2013	Plan 2013	Plan 2014
EUR	Hauser						1-12
Cost Element							
420000	Fertigungs-Loehne	1	314.174	313.902	94.113	313.902	1.200,00
421000	Hilfs-Loehne	1					1.837,00
466000	Versicherungen	4	3.938	4.008	995	4.008	11.695,89
#	Löhne	EG	318.112	317.910	95.109	317.910	14.732,89

Abbildung 11-1: StyleSwitch - ohne Schreibschutz

H1410	Dienstleistungen	EC	Ist 2012	Plan 2012	IST 1-003 2013	Plan 2013	Plan 2014
EUR							1-12
Cost Element							
420000	Fertigungs-Loehne	1	487.020	486.197	152.662	486.197	21.203,27
421000	Hilfs-Loehne	1					3.537,00
466000	Versicherungen	4	76.575	77.112	19.355	77.112	46.783,56
#	Löhne	EG	563.595	563.309	172.017	563.309	71.523,83

Abbildung 11-2: StyleSwitch – with write protection

Macro

The function call is made with "StyleSwitch;[name of styleswitch]", e.g. "StyleSwitch;Lock", when the macro is executed, all lines with the same name are executed.

Configuration table

StyleSwitchSettings

Name	Style	CellR	CellG	CellB	Lock
Lock	KernWrite*	252	252	247	1
Lock	KernComment	252	252	247	1
Lock	KernJumpPlan	252	252	247	1
Unlock	KernWrite*	255	255	204	0
Unlock	KernComment	255	255	204	0
Unlock	KernJumpPlan	85	183	255	0

Abbildung 11-3: StyleSwitchSettings

Setting	Description
Name	The name can occur several times; when the macro is executed, all lines are executed with the same name.
Style	The placeholder "*" can also be used to change several format templates. "KernWrite*" finds KernWrite, KernWrite0, etc.
CellR, CellG, CellB	Color code to which the background color is to be changed; values from 0-255 are permitted.
Lock	Empty or 0 - (default) write protection is not active 1 - Write protection is activated

Note:	In the StyleSetDefinitions table of the StyleCorner, you can use the "Protection" field to define a format template that differs from the format template.

11.6 Additional STYLExx-Pointer (from 4.3.2)

From version 4.3.2, additional pointers in the format STYLE01, STYLE02, etc. can be set in addition to STYLE. This makes it possible to carry out different formatting, for example for read/write and totals lines, without having to create an additional StyleCorner.

Style01

	STYLE01	STYLE02	
STYLE01			
STYLE02			

To do this, an additional Pointer column must be inserted in the StyleSetDefinitions table.

12 Formula

Formula01

The Formula module is used to copy formulas from the header area to the relevant rows. These formulas can be e.g. cost element specific and the formula results can also be inserted as values.

12.1 Brief overview

Area	Feature	Info	
Code			
VBA	basKitFormula		
	clsKitFormula		
Settings			
Configuration sheet	Formula		
Setting tables	FormulaAreas		
	FormulaKeyMatch		
Controls			
Macro	FormulaActivate		
	FormulaActiveInLine		
Cado			
UserExits			
Impact Area			
Corner	FormulaXX		
KeyPointer	FORMULA	=formula	
	ROW / RowAll	X	First row of the model space
	KEY (optional)	An entry of the key column from [FormulaKeyMatch].	
	FORMULAKEYMATCH	A column header in [FormulaKeyMatch].	
	FORMULATOVALUE	1	Converts formulas to values

12.2 Settings

The corner is set up as usual: {FormulaXXColumn} and {FormulaXXRow}. Via the table [FormulaAreas] on the worksheet |Formula| the desired FormulaCorner are activated.

FormulaAreas

Id	Active
Formula01	1
Formula02	1
Formula03	1

Table FormulaAreas

With the macro *FormulaActivate* all FormulaCorner of the table are executed. The macro must be entered in a UserExit on the customizing sheet. If you want to execute single corners at different times (e.g. AFT_LEAD_SAT, AFT_READ) you can add this project specific.

It is also possible to execute only one specific corner.

Multiple corners can be specified - separated by "|".

Customizing

Process	SubProcess	Parameter	Key	Description	Value
UserExit	Macro	AFT_READ		Execute Formulas	FormulaActivate
UserExit	Macro	AFT_READ		Execute Formulas	FormulaActivate Formula01|Formula02
UserExit	Macro	AFT_READ		Execute Formulas	FormulaActivate Formula01

Macros in the customizing sheet

To use the Formula, FormulaToValue and FormulaKeyMatch pointers only on the current page (e.g. MultiPage) and only in the active line(s), the FormulaActivateInLine function can be used (e.g. in combination with the PickList or via the menu ribbon).

Custom Button

Custom Button					
SheetName	View	Icon	MacroName	Caption	Tooltip
ExampleFormula		ProjectBranchEngineering	FormulaActivateInLine Formula02	FormulaInLine	

Macros as CustomButton in the navigation

12.3 Controls

Formula01

	ROW	KEY				
FORMULA				=100	=200	
	X					
		1		=100	=200	
		2		=100	=200	
		3		=100	=200	
		5		=100	=200	
		6		=100	=200	

FormulaCorner with KeyPointer

FORMULA

Marks the row where the formulas to be copied must be entered.

ROW / RowAll

In the KeyPointer *Row the X* marks the row below which the formulas are copied into.

Note:	When using the FormulaCorner with satellites in *All* mode, the corresponding KeyPointer is called *RowAll*.

KEY (optional)

Below the KeyPointer *key,* additional rows can be marked into which the formulas are to be copied. The check is made for <> Empty. If the KeyPointer *Key is* not specified, the formulas are copied into the entire column starting from Row/X.

FORMULAKEYMATCH

The KeyPointer *FormulaKeyMatch* is used to copy e.g. cost element specific formulas. For this purpose, any number of formulas are entered in the [FormulaKeyMatch] table for each key (=row key, cost element...). The column identifier *Key* is fixed, all others can be freely selected.

FormulaKeyMatch

Key	Formula1	Formula2	Formula3
AAA	=1	=10	=100
BBB	=2	=20	=200
CCC	=3	=30	=300
DDD	=4	=40	=400

FormulaKeyMatch table

Formula02

	ROW	KEY				
FORMULAKEYMATCH				Formula1	Formula2	Formula3
	X					
		AAA		=1	=10	=100
		DDD		=4	=40	=400

FormulaCorner with FormulaKeyMatch

FORMULATOVALUE

The (optional) KeyPointer *FormulaToValue* causes formula results to be converted to values. The corresponding columns are marked with the number *1*.

Formula03

	ROW	KEY					
FORMULA				=100	=200		
FORMULAKEYMATCH						Formula1	Formula2
FORMULATOVALUE					1		1
	X						
		AAA		=100	200,0	=1	10,0
		DDD		=100	200,0	=4	40,0

FormulaCorner with "FormulaToValue

12.4 SumActivate and SumReverseActivate

The FormulaCorner replaces the startSum or startSumReverse sum generation of the StructureBuilder. In contrast to the StructureBuilder, the FormulaCorner does not require virtual sum levels.

Note on the sum generation	The formula =SUM only allows 255 parameters. For very large groups, this value may be exceeded. In this case, the =SUMIF formula is used.

Pointer:

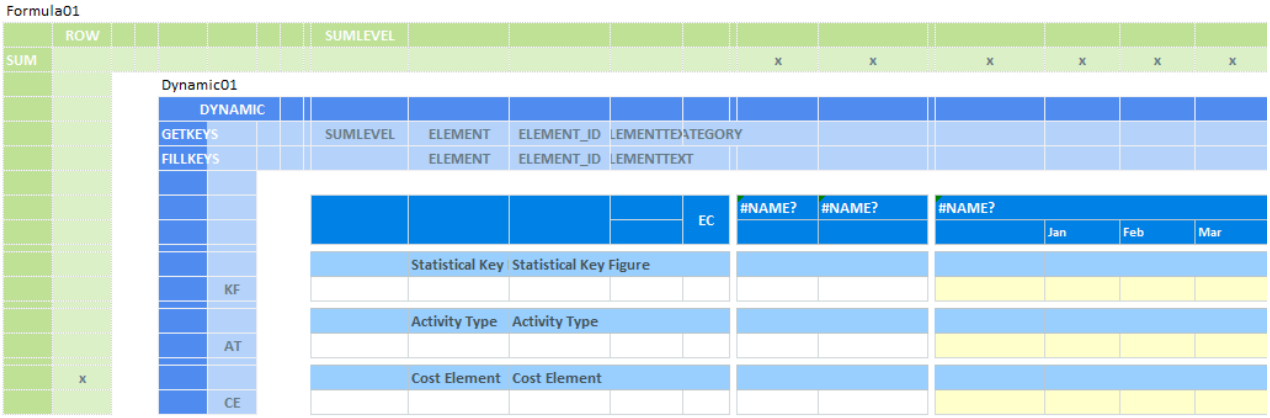
- SUMLEVEL: marks the column with the levels (only numerical values allowed, i.e. not ## or ***).
- SUM: use "x" to mark the columns in which the totals are to be entered.
- ROW: as usual, mark the row with "x" under which the summation is to begin.

Function calls:

- SumActivate (replaces startSum). The corner name can be given as a parameter when calling (as here).
- SumReverseActivate (replaces startSumReverse). Parameters like SumActivate.

Like startSum, the methods can be called via UserExit/menu ribbon etc.

The pointers for the FormulaActivate/FormulaActivateInLine functionality can be created in the same corner. As there are separate function calls, the time at which the functions are called (e.g. via UserExit) can be varied.



13 Stretch

Stretch01

		COPY							
STRETCH									
REFERENCE									
TOEXCELROW									

Stretch brief overview

Area	Feature	Info
Code		
VBA	basKitStretch	
	clsKitStretch	
Settings		
Configuration sheet	Stretch	
Setting tables	StretchAreas	
	FormulaKeyMatch	
Controls		
Macro	StretchStart	
Cado	CadoBeforeStretch	CadoBeforeStretch is executed per active stretch corner and gets the active corner as parameter with
UserExits	AFT_STRETCH	Macro and View
	AFT_SHRINK	Macro and View
Impact Area		
Corner	StretchXX	
KeyPointer	STRETCH	marks corners that are to be stretched
	REFERENCE	Marks the reference column
	TOEXCELROW	(optional) Excel Row up to which is extended
	COPY	This row is duplicated

13.1 Controls

With the macro *StretchStart* all StretchCorner of the table are executed. The macro must be entered at a UserExit.

Note:	If you want to execute individual corners at different times (e.g. AFT_LEAD_SAT, AFT_READ), this would have to be added on a project-specific basis.

13.2 Settings

The corner is set up as usual: {StretchXyColumn} and {StretchXyRow}. Via the table [StretchAreas] the desired StretchCorner are activated:

StretchAreas

Id	Active
Stretch01	1

KeyPointer of the Stretch Corner

STRETCH

All corners that are to be extended must be marked here with the corresponding reference character (always the first column of the corner, see example below)

If a corner (e.g. *NavigationCorner*) is to be extended to the maximum existing corner length, either all references can be specified, of which the highest value is to be used, or *, if the highest value of all existing references is to be used (see example). In the case it is important that in row alignment also the start row is marked with *.

REFERENCE

Here you can mark the columns from which a structured table or the data area starts. The marker character (in the example below A and B) is used as reference to mark the first columns of the corner (see pointer *STRETCH*)

TOEXCELROW (optional)

The corner range extension can be extended either to a defined row or to the end of an existing structured table.

In the first case, the (Excel) row can be entered here in the column up to which the corner areas are to be extended. Here the value must be entered in the same column in which a reference is also present, see *REFERENCE* and example below. If a value is entered, it is always extended up to this row, regardless of whether a structured table is present.

If no value is entered, the ranges are extended to the end of the table if a structured table exists.

IMPORTANT: It is NOT the number of rows in the table!!!

COPY

Together with *REFERENCE,* marks a cell located within the structured table to be used as a reference.

The row also serves as (start) copy row, is marked here per reference with the corresponding character. If the start row is the same for several, all characters can be entered interlinked (e.g. "AB") or only with "*" for all areas.

Only formulas are copied, constants are not copied. This prevents that for example "*" from the KeyPointer *FREEZE* is copied.

If you want to copy a constant, you can use a pseudo formula, e.g.: *A* becomes *="A"*

Note:	The following formula can be used to determine the last row of a structured table, for comparison see dynamic name ranges. **=SUM(** **ROW(ST[#Headers]);** **ROWS(ST))** (ST is the name of the structured table)

14 Splasher

Splasher01

	DIMENSION	CATEGORY	ELEMENT	SUMLEVEL	ROW				
YEAR									
MONTH									
DIMENSION									

Allevo has functions for converting annual data into the associated monthly values and vice versa ("horizontal distribution"). Alternatively, "vertical distribution" is also possible, e.g. to automatically distribute input at group level to the associated cost elements. The reference values required for the distribution can be stored flexibly in the master.

Brief overview

Area	Feature	Info
Code		
VBA	basKitSplasher	
	clsKitSplasher	
	clsKitSplasherVariants	
Settings		
Configuration sheet	Splasher	
Setting tables	SplasherAreas	
	SplasherCommonSettings	
	SplasherVariantSettings	
	SplasherDistKeyChoices	
	SplasherDistKeyBaseCurve	
	SplasherReferenceCurve	
Controls		
Macro	SplasherGenerateSums	
	SplasherPeriodToYear	
	SplasherYearToPeriod	
	SplasherActivate	
	SplasherDeactivate	
Cado		
UserExits		
Impact Area		

Corner	SplasherXX		
KeyPointer in column alignment	YEAR	any, optionally with suffix	
		_DK	Distribution Key
		_Fixed	Annual value
		_variable	vertical reference distribution
	MONTH	like Year, without suffix	
	DIMENSION	Q	Quantity
		V	Value
		X	Q+V
	SUMLEVEL	1	Columns for Vertical Splasher
KeyPointer in row a-lignment	ROW	X	First row of the model space
	CATEGORY	Element Category for [SplasherVariantSettings].	
	DIMENSION	Q	Quantity
		V	Value
	SUMLEVEL	Sum stage for vertical splasher	
	ELEMENT	Element column for "analog FTE	

14.1 Controls

The splasher can be started or executed in two ways:

Call the initial splasher macros via UserExit:

- SplasherGenerateSums
- SplasherPeriodToYear
- SplasherYearToPeriod

Optionally, a specific SplasherCorner can be given as a parameter to the three macros mentioned. The transfer takes place separated by semicolons and without spaces.

Example for the UserExit macro: SplasherGenerateSums; Splasher01

If no SplasherCorner is specified, all splashers activated in the [SplasherAreas] table are initially splashed in sequence.

Input in Excel within the sensitive splasher range:

The prerequisite for this is the activation of the splasher:

The splasher activity is controlled by two macros in addition to the activation of the SplasherCorner in the [SplasherAreas] table:

- SplasherActivate
- SplasherDeactivate

These can be entered in the UserExits.

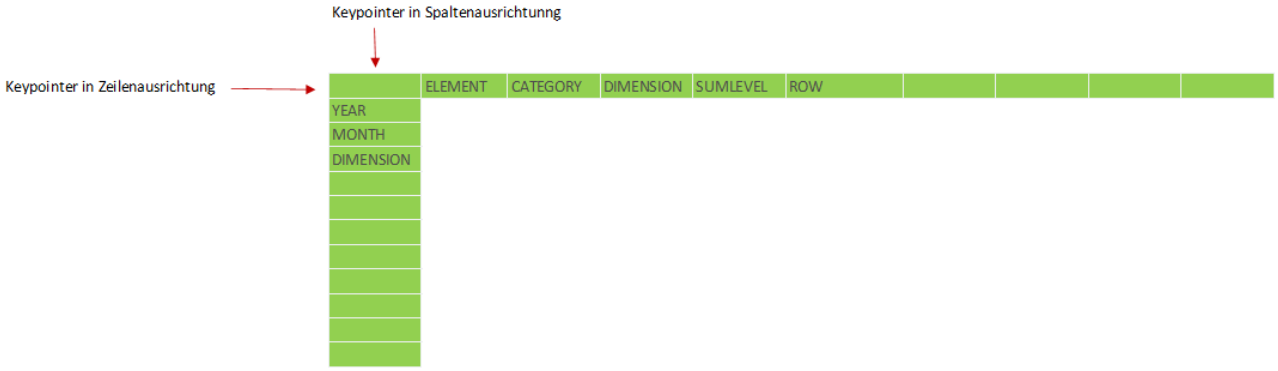
If the splasher is active, each entry in Excel is checked to see if the value entered is in a range relevant to the splasher (splasher sensitive range).

The sensitive area is determined based on the configuration in SplasherCorner and in the Splasher configuration tables.

14.2 Settings

The corner is set up as usual with the two areas {SplasherXYColumn} and {SplasherXYRow}.

KeyPointers in column and row alignment must be entered.



KeyPointer in column alignment:

YEAR

In this row the columns with annual plan values are marked; The marker pointer of the respective variant is defined in the [SplasherVariantSettings] table.

MONTH

In this row the month columns are marked. Marker pointer must correspond to the associated year column variant.

Any number of variants can be defined (e.g. actual planning (variant *A*), forecast (variant *B*)). The variant names can be arbitrary, they only have to be identical with the names in the table [SplasherVariantSettings] (please also pay attention to space characters). Depending on the varianta year column and a month column must be defined. The month columns do not have to be contiguous.

DIMENSION

In this row, the dimension of the annual plan values is assigned in column alignment *Q*, *V* or *X*; where *Q stands* for Quantity and *V* for Value. *X* applies to *Q* and *V.* The associated monthly values are determined according to the dimension specified in row alignment (see description of the KeyPointers in row alignment).

From 4.3.2	From version 4.3.2, 1,2,3 can also be used instead of Q,V,X. To do this, the setting "NumericDimension" and the value 1 must be entered in the SplasherCommonSettings.

SUMLEVEL

In this row, the columns that allow input via vertical splasher are marked with a *1*: After vertical splashing, all sum levels in these columns are calculated via *SplasherGenerateSums.*

	CATEGORY	DIMENSION	SUMLEVEL	ROW										
YEAR						A	A							
MONTH								A	A	A	A	A	A	A
DIMENSION						Q	V							

KeyPointer in row alignment

	CATEGORY	DIMENSION	SUMLEVEL	ROW
YEAR				
MONTH				
DIMENSION				
				X
	1	Q	0	
		Q	1	
	1	V	0	
	1	V	0	
	1	V	0	
		V	1	

CATEGORY

In this column, a category (cost element category) is entered for each relevant plan row. The categories must be defined for each variant in the [SplasherVariantSettings] table to enable the plan row for the respective variant. The categories are separated with "|". If no or no defined category is entered, this row is locked for the splasher. (Note: The category "A" stands for "Average").

SplasherVariantSettings

Variant	Perito	AllowedCategories
A	3	1|4|3|A|S|11|12
B		1|4|S|A|11|12

SUMLEVEL

For the vertical splasher the specification of the sum hierarchy levels is necessary. This is specified in row alignment in the SUMLEVEL column. Both numeric sum levels and characters are possible, whereby this must be defined in the [SplasherCommonSettings] setting table. For numeric summation level representation the setting for *sumFlag* is set to 1. If the summation levels are to be displayed with e.g. '#', then # must be entered as setting.

SplasherCommonSettings

Common	Setting
activationByStatus	0
decimals	2
writeZero	1
CheckSettings	1
horizontal	1
vertical	0
sumFlag	1
AllSumLevels	0

ROW

Analogous to the other Corners, in the ROW column an "X" marks the row from which the plan area is to begin. Below "X" the splasher area would start.

ELEMENT **(Optional)**

The cost elements are entered in this column. These can then be referenced, for example, with *analog FTE.*

SplasherDistKeyBaseCurve

Distribution	Variant	Table	DisKey	Default	Element
End of Quarter		SplasherReferenceCurve	EOQ		
Halfyear		SplasherReferenceCurve	HALFYEAR		
analog FTE	B				FTE
analog FC	B				

Splasher configuration tables

The splasher requires the three configuration tables [SplasherAreas], [SplasherCommonSettings] and [SplasherVariantSettings] already mentioned. If splashing with individual distribution is desired, the table [SplasherDistKeyBaseCurve] is additionally required (see the explanations in Splasher Features).

SplasherAreas

SplasherAreas

Id	Active
Splasher99	1
Splasher01	0

Only the SplasherCorner that have been entered and activated in this table will be executed (1=active, 0=inactive).

SplasherCommonSettings

SplasherCommonSettings

Common	Setting
activationByStatus	0
decimals	2
writeZero	1
CheckSettings	1
horizontal	1
vertical	0
sumFlag	1
AllSumLevels	0

These settings are valid for all defined SplasherCorner and are read once. If settings are changed during a running session, the *SplasherActivate* macro must be executed so that the settings are read again.

Common	Setting		Meaning
activationByStatus	1= active/0= inactive		Value
horizontal	1= active/0= inactive		Enable/disable horizontal splashing
vertical	1= active/0= inactive		Enable/disable vertical splashing
sumFlag	1	Sum level as numeric value	for the use of numerical sums (1-99)
	#	Sum level as number of # characters	
	*	Sum level as number of * characters	
decimals	0-n		Number of decimal places for filling the automatically generated values.

writeZero	1= active/0= inactive	Create 0 as entry for deleted values (useful to mark cancel-lations)
AllSumLevels	1= active/0= inactive	Activation of the vertical splasher for summation levels > 1

SplasherVariantSettings

SplasherVariantSettings		
Variant	Perito	AllowedCategories
A	3	1|4|3|A|S|11|12
B		1|4|S|A|11|12

In this table, *Perito* and *AllowedCategories* are defined per Variant.

Perito can be used to "lock" months that have already been scheduled. These are no longer overwritten.

The categories are used to define plan rows, which must be specified in row alignment in the corner in the *CATEGORY column* (usually the column with the *ElementCategory* is referenced here).

Splasher features

Horizontal distribution options

Horizontal distribution is enabled by activating the setting *horizontal* in the [SplasherCommonSettings] table.

The default distribution of the splasher is determined by checking already existing distribution. If there are no monthly values yet, the annual value is distributed linearly. Otherwise, the existing distribution curve is kept.

Optionally, one column with corresponding distribution key can be displayed for each variant.

For this purpose, the pointer *Variantname_DK* must be entered in any column of the KeyPointer YEAR row. The variant identifier *A* thus becomes *A_DK*.

	CATEGORY	DIMENSION	SUMLEVEL	ROW								
YEAR						A	A	A_DK				
MONTH									A	A	A	A
DIMENSION						Q	V		X	X	X	X
				X		Year Qty	Year Value	DisKey	M1	M2	M3	M4
	1	Q	0			1200		linear	100	100	100	100
		Q	1			**1200**		**linear**	**100**	**100**	**100**	**100**
	1	V	0				1400	manually	100	300	100	100
	1	V	0				0	linear	0	0	0	0
	1	V	0				0	linear	0	0	0	0
		V	1				**1400**	**manually**	**100**	**300**	**100**	**100**

The current distribution key is then displayed per row:

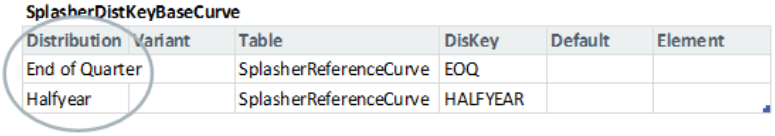
If the current distribution curve is maintained, *manual is* automatically entered and linear if the distribution is *linear*:

Use of distribution curves (seasonalization)

Individual distribution curves can be defined for the splasher (e.g. quarterly or half-yearly distribution of annual values). The desired distribution keys can be defined in the [SplasherDistKeyBaseCurve] table.

Example:

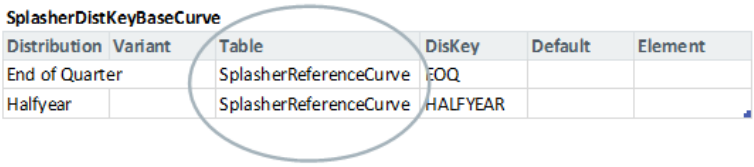
1. Define distribution curve designation (names can be freely selected). This designation is evaluated by the splasher in the *Variant_DK* splasher column.



SplasherDistKeyBaseCurve

Distribution	Variant	Table	DisKey	Default	Element
End of Quarter		SplasherReferenceCurve	EOQ		
Halfyear		SplasherReferenceCurve	HALFYEAR		

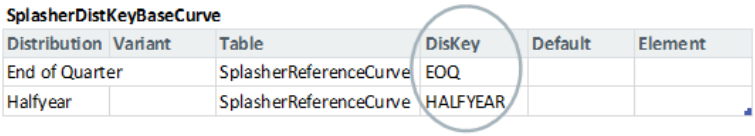
2. Specify table in which the distribution curves are stored.



SplasherDistKeyBaseCurve

Distribution	Variant	Table	DisKey	Default	Element
End of Quarter		SplasherReferenceCurve	EOQ		
Halfyear		SplasherReferenceCurve	HALFYEAR		

3. Define distribution key designation in the specified table



SplasherDistKeyBaseCurve

Distribution	Variant	Table	DisKey	Default	Element
End of Quarter		SplasherReferenceCurve	EOQ		
Halfyear		SplasherReferenceCurve	HALFYEAR		

4. Create the specified structured table with the distribution curves

The table must have the following column labels: | Diskey | 1 | 2 |... | n |

The number of months can be arbitrary, but should of course correspond to the period as defined in the splasher!

SplasherReferenceCurve

Diskey	1	2	3	4	5	6	7	8	9	10	11	12
HALFYEAR			0,5						0,5			
EOQ			0,25			0,25			0,25			0,25

The diskey column of the table must contain the diskey names defined in the [SplasherDistKeyBaseCurve] table. They are protected against incorrect entries via data validation with selection box.

Example in the Splasher Corner:

	CATEG	DIMEN	SUMLE	ROW														
YEAR						A	A	A_DK										
MONTH									A	A	A	A	A	A	A	A	A	A
DIMENSION						Q	V		X	X	X	X	X	X	X	X	X	X
				X		Year Qty	Year Value	DisKey	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12
	1	Q	0			1200		linear	100	100	100	100	100	100	100	100	100	100	100	100
		Q	1			**1200**		**linear**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
	1	V	0				1400	End of Quart	0	0	350	0	0	350	0	0	350	0	0	350
	1	V	0				3000	linear	250	250	250	250	250	250	250	250	250	250	250	250
	1	V	0				5000	Halfyear	0	0	2500	0	0	0	0	0	2500	0	0	0
		V	1				**9400**	**manually**	**250**	**250**	**3100**	**250**	**250**	**600**	**250**	**250**	**3100**	**250**	**250**	**600**

Set up analog distribution options

Analogous to an existing distribution in a given row:

The splasher can distribute annual values analog to an existing distribution in the splasher. For this purpose, the variant according to which the distribution is to be made must be specified in the [SplasherDistKeyBase-Curve] table. In addition, an element can be entered to specify a specific distribution of the analog variant.

Example: Distribution of the annual value (Variant A) analogous to Variant B, element FTE

Note: For this option, the KeyPointer *ELEMENT* must be created in the SplasherCorner (in row alignment, see figure below), if it does not already exist. The cost elements should then be entered in this column. The KeyPointer can be in any column in row alignment (i.e. first row of the {SplasherXYColumn} area).

SplasherDistKeyBaseCurve

Distribution	Variant	Table	DisKey	Default	Element
End of Quarter		SplasherReferenceCurve	EOQ		
Halfyear		SplasherReferenceCurve	HALFYEAR		
analog FTE	B				FTE
analog FC	B				

	ELEMENT	CATEGO	DIMEN	SUMLE	ROW													
YEAR						A	A_DK					B	B_DK					
MONTH								A	A	A				B	B	B		
DIMENSION						V						V						
					X													
	FTE	1	V	0		1200	linear	400	400	400		200	manually	50	50	100		
		1	V	1		**1200**	**linear**	**400**	**400**	**400**		**200**	**manually**	**50**	**50**	**100**		
	400000	1	V	0		1500	analog FC	300	600	600		250	manually	50	100	100		
	420000	1	V	0		2700	analog FTE	675	675	1350		900	linear	300	300	300		
	430000	1	V	0		300	linear	100	100	100		1200	linear	400	400	400		
		1	V	1		**4500**	**manually**	**1075**	**1375**	**2050**		**2350**	**manually**	**750**	**800**	**800**		

Analogous to an existing distribution in the same row

If no specification is made in the element column of the [SplasherDistKeyBaseCurve] table, the distribution is made analogous to the specified variant in the same row of the year value to be splashed.

Example: Distribution of the annual value (Variant A) analogous to Variant B (analogous to the forecast distribution)

SplasherDistKeyBaseCurve

Distribution	Variant	Table	DisKey	Default	Element
End of Quarter		SplasherReferenceCurve	EOQ		
Halfyear		SplasherReferenceCurve	HALFYEAR		
analog FTE	B				FTE
analog FC	B				

A	A_DK					B	B_DK				
		A	A	A				B	B	B	
V						V					
1200	linear	400	400	400		200	manually	50	50	100	
1200	**linear**	**400**	**400**	**400**		**200**	**manually**	**50**	**50**	**100**	
1500	analog FC	300	600	600		250	manually	50	100	100	
2700	analog FTE	675	675	1350		900	linear	300	300	300	
300	linear	100	100	100		1200	linear	400	400	400	
4500	**manually**	**1075**	**1375**	**2050**		**2350**	**manually**	**750**	**800**	**800**	

Use of a defined default distribution

In case the selected distribution is equal to zero and the standard horizontal distribution logic of the splasher should not take effect or no distribution takes place, a default distribution can be defined in the [SplasherDistKeyBaseCurve] table for each variant. For this, the variant must be specified in the default column for the desired distribution. Several entries are considered separately: e.g. entry "AB" would mean that this distribution is drawn for (A) as well as for (B). Optionally, it is also possible to work with "*".

SplasherDistKeyBaseCurve

Distribution	Variant	Table	DisKey	Default	Element
analog FC	B				
End of Quarter		SplasherReferenceCurve	EOQ	B	
Halfyear		SplasherReferenceCurve	HALFYEAR	A	

Horizontal distribution of sums

Yearly totals can also be distributed horizontally using the above options. The distribution key of the totals is decisive for the entire associated block.

For example, if the sum is distributed 'linearly', all distribution keys in the associated block are automatically set to linear and distributed linearly.

SUMLEVEL	W				B	B_DK												
							B	B	B	B	B	B	B	B	B	B	B	B
					V		X	X	X	X	X	X	X	X	X	X	X	X
	X			Year Val	DisKey	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	
0			1000		manuall	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	
1			**1000**		**manuall**	**1000**	**1000**	**1000**	**1000**	**1000**	**1000**	**1000**	**1000**	**1000**	**1000**	**1000**	**1000**	
0			1200	1200	linear	100	100	100	100	100	100	100	100	100	100	100	100	
0		Block	1200	1200	linear	100	100	100	100	100	100	100	100	100	100	100	100	
0			1200	1200	linear	100	100	100	100	100	100	100	100	100	100	100	100	
1			**3600**	**3600**	**linear**	**300**	**300**	**300**	**300**	**300**	**300**	**300**	**300**	**300**	**300**	**300**	**300**	

Horizontal distribution of monthly values with fixed annual value

By default, the annual value is adjusted according to the changed monthly values.

Optionally, the year value can be fixed and the month distribution can be adjusted. For this purpose, the pointer *Variant_Fixed* can be entered within the keypointer area YEAR for each variant.

For each row, if the *Variant_Fixed* pointer is present, a check is made to see whether a specification has been made (i.e. if the content of the cell<>empty => fixed year value). Any specifications (e.g. "x" or "fix") can be made here.

Example: The splasher sets the distribution key to *manually* after the distribution with fixed year.

	ELEMENT	CATE	DIME	SUMLEV	ROW															
YEAR						B_FIXED	B	B	B_DK											
MONTH										B	B	B	B	B	B	B	B	B	B	B	B
DIMENSION							Q	V		X	X	X	X	X	X	X	X	X	X	X	X
					X	Year Fixed	Year Qty	Year Val	DisKey	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12
	FTE	A	Q	0			1000			1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
			Q	1			**1000**		**manually**	**1000**	**1000**	**1000**	**1000**	**1000**	**1000**	**1000**	**1000**	**1000**	**1000**	**1000**	**1000**
		1	V	0		x		1200	manually	104,55	50	104,55	104,55	104,55	104,55	104,55	104,55	104,55	104,55	104,6	104,5
	420000	1	V	0		fix		2400	manually	190,91	300	190,91	190,91	190,91	190,91	190,91	190,91	190,91	190,91	190,9	190,9
	430000	1	V	0				40000	linear	3333,33	3333,33	3333,33	3333,33	3333,33	3333,33	3333,33	3333,33	3333,33	3333,33	3333	3333,4
			V	1				**43600**	**manually**	**3628,79**	**3683,33**	**3628,79**	**3628,79**	**3628,79**	**3628,79**	**3628,79**	**3628,79**	**3628,79**	**3628,79**	**3629**	**3628,8**

Vertical distribution options

Vertical distribution is enabled by activating the setting *vertical* in the [SplasherCommonSettings] table (see above: Explanation of splasher configuration tables).

1. Automatic summation of the element blocks (years and months) according to the sum hierarchy levels up to the 1st hierarchy level when changing the year or month values. Existing formulas are not overwritten.

2. Distribution of the sum within the associated block to the individual elements:

If a sum is adjusted manually, the sum is distributed vertically to the individual elements. Vertically, the existing distribution is retained. In addition, horizontal splashing is performed (synchronization of monthly or annual values).

Note:	If the *AllSumLevels* setting is activated, splashing can be performed over several sum levels. However, caution is advised here: - The element hierarchy must be consistent, only sums may lie under sum levels > 1 - When splashing over multiple summation levels, the performance requirement increases exponentially

Vertical reference distribution

A reference distribution can be defined for the vertical distribution of a sum per variant.

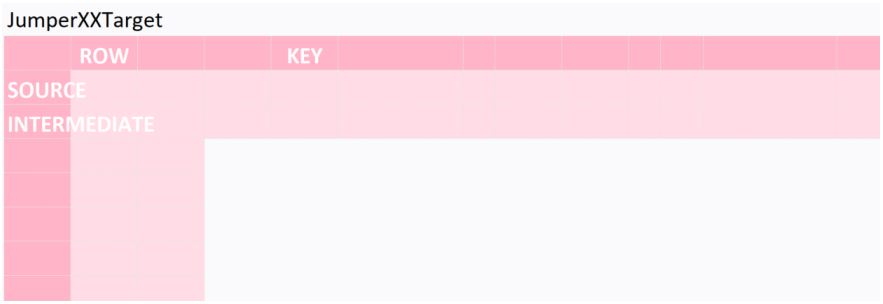
For this the pointer *Variant_VERTICAL* must be entered within the keypointer area *YEAR.*

Example:

	ELEMENT	CAT	DIME	SUM	ROW																
YEAR						B_VERTICAL	B	B	B_DK												
MONTH										B	B	B	B	B	B	B	B	B	B	B	B
DIMENSION							Q	V		X	X	X	X	X	X	X	X	X	X	X	X
					X		Year	Year Value	DisKey	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12
	FTE	1	Q	0			120		linear	10	10	10	10	10	10	10	10	10	10	10	1
			Q	1			**120**		**linear**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**1**
		1	V	0			0	0	linear	0	0	0	0	0	0	0	0	0	0	0	
	420000	1	V	0			1	4000	manually	333,33	333,33	333,33	333,33	333,33	333,33	333,33	333,33	333,33	333,33	333,33	333,3
	430000	1	V	0			0	0	linear	0	0	0	0	0	0	0	0	0	0	0	
			V	1				**4000**	**manually**	**333,33**	**333,33**	**333,33**	**333,33**	**333,33**	**333,33**	**333,33**	**333,33**	**333,33**	**333,33**	**333,33**	**333,3**

The vertically distributed sum is then splashed horizontally. The distribution key is not affected by the vertical splashing, since it only acts horizontally.

15 Jumper



The aim of the jumper is to copy data from a source (usually the planning mask) into one or more rows of a structured target table (usually a satellite) by double-clicking. The number of rows to be appended to the table can either be fixed, specified by the user, or determined dynamically via an intermediate table.

Brief overview

Area	Feature	Info
Code		
VBA	basKitJumper	
	frmKitJumper	
	clsKitJumper	
	clsKitJumperDialog	
	clsKitJumperSource	
	clsKitJumperIntermediate	
	clsKitJumperTagret	
Settings		
Configuration sheet	Jumper	
Setting tables	JumperAreas	
Controls		
Macro	JumperActivate	Called by the Excel event "Workbook_SheetBeforeDoubleClick
Cado		
UserExits		
Impact Area		
Corner	JumperXXSource	
	JumperXXTarget	
KeyPointer: Source	ROW	Marks the first row of the effective area
	X_INSERT	Rows to be transferred to the target.
	X_INSERTED	Here, VBA enters a 1 after the transfer to the target.
	TARGET	Fields whose contents are to be transferred to the target table.

	SENSITIVE	Fields to which the double-click should react
KeyPointer: Target	ROW	Marks the first row of the effective area
	KEY	Consecutive row number generated by VBA
	SOURCE	Fields whose contents are taken from the source
	SORT	(optional) sorting after expanding the table
	INTERMEDIATE	(optional) Controls the inclusion of a third table
	FILL	(optional) number of rows of the target in which source values are entered

15.1 Controls

The *JumperActivate* macro cannot be called via UserExits or similar. It is always activated by the Excel event ***Workbook_SheetBeforeDoubleClick.***

15.2 Settings

The desired jumper corners are activated via the [JumperAreas] table:

JumperAreas

Id	Active
Jumper01	1

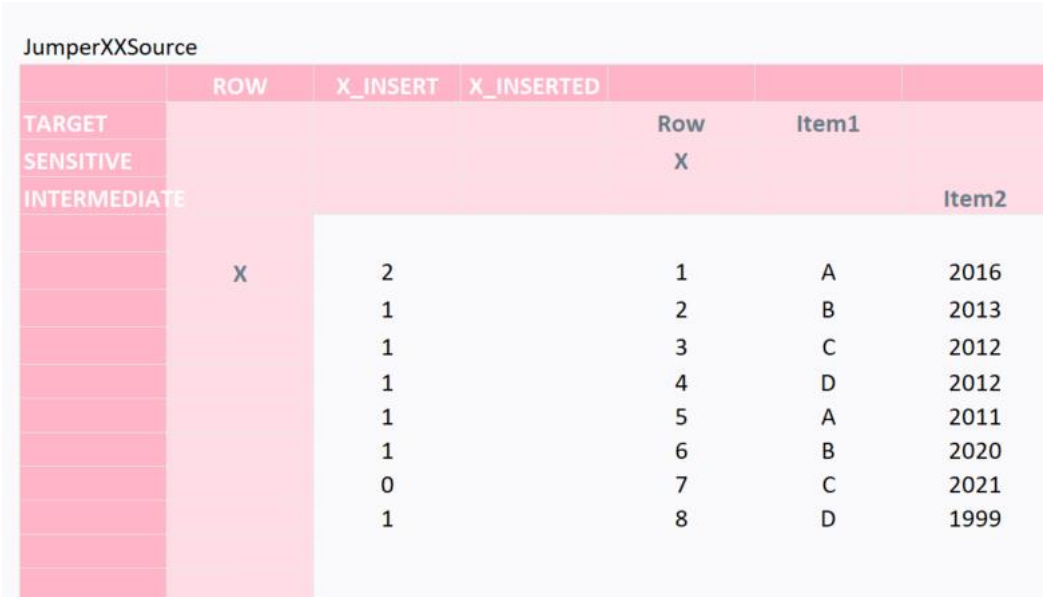
Set up source

The following pointers are mandatory:

- TARGET: Fields whose contents are to be transferred to the target table.
- SENSITIVE: One or more fields to which the double-click should react (everything except empty allowed, here: X).
- ROW: The x marks the row below which the data range begins.
- X_INSERT: The entries can be used to control which rows are to be transferred to the target.
- X_INSERTED: Here VBA enters a 1 after the double-clicked row has been transferred to the target.

Optional INTERMEDIATE pointer

INTERMEDIATE: When working with an intermediate table, the source pointer INTERMEDIATE can be created (optional). If the pointer SOURCE is set up in Intermediate, then the rows are only created in the target if the field contents of SOURCE | INTERMEDIATE and INTERMEDIATE | SOURCE match.



Active rows

Whether a double-clicked row is considered by the jumper depends on the entries at X_INSERT and X_INSERTED:

wird typischerweise per Formel gesetzt

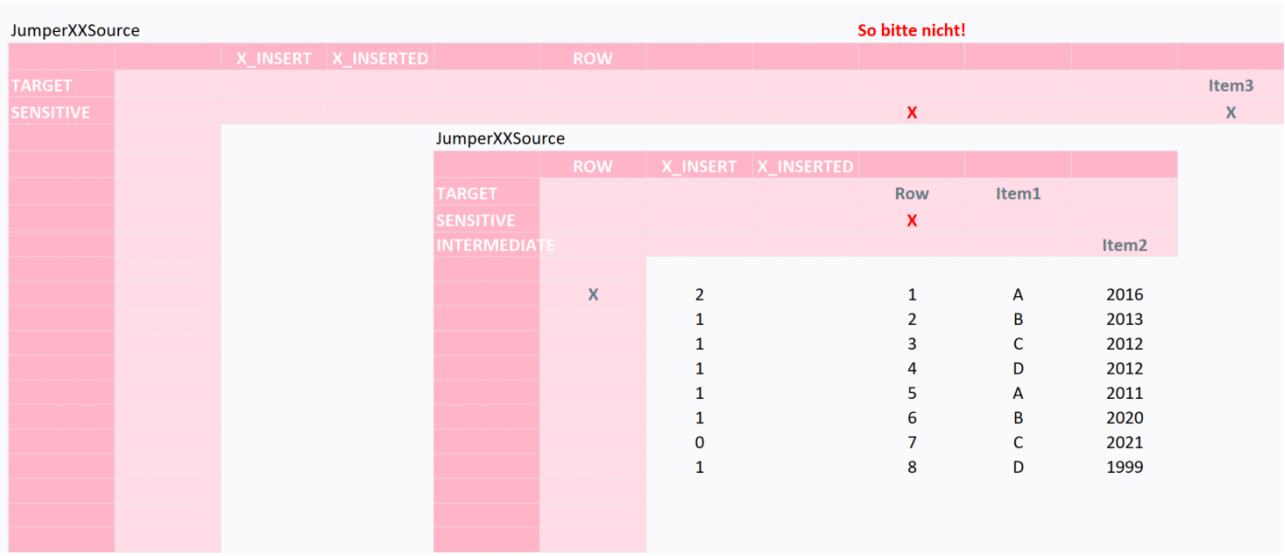
diese Spalte darf keine Formel beinhalten, da sie von VBA befüllt wird

X_INSERT	X_INSERTED	Action
<blank>	<blank> | n	nein
0	<blank> | n	nein
1	<blank> | 0	ja
1	1	nein
2	<blank> | 0 | 1	ja

If a row was transferred in the target, VBA sets "1" in field X_INSERTED of the double-clicked row.

Interleave multiple jumpers

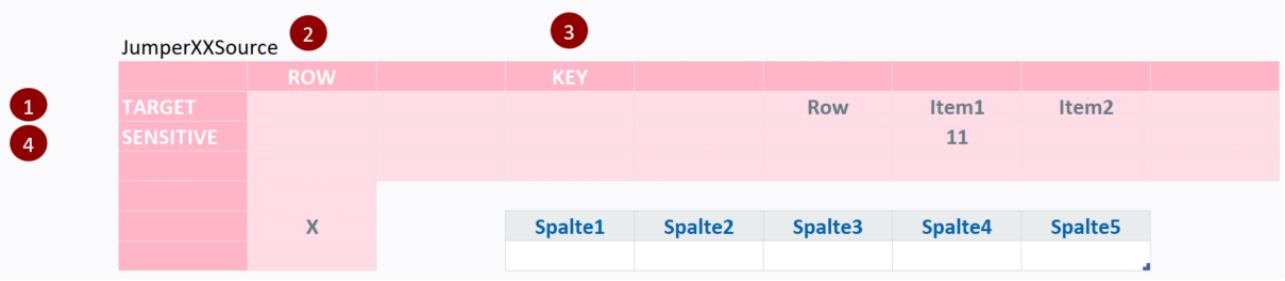
If several jumper source corners are superimposed, the pointers for SENSITIVE must not be in the same column, otherwise VBA cannot clearly determine which corner must be served.



Set up target

The {JumperXyTargetColumn} or {JumperXyTargetRow} areas are set up for the target corner.

Target always has a structured table. This is extended by the determined rows. So no whole Excel rows are inserted. For this reason a "trunk corner" is sufficient (as e.g. with the satellite All).



Source pointer

The following pointers are mandatory:

- TARGET: Fields whose contents are to be transferred to the target table.
- SENSITIVE: One or more fields to which the double-click should react (everything except empty allowed, here: X).
- ROW: The x marks the row below which the data range begins.
- X_INSERT: The entries can be used to control which rows are to be transferred to the target.
- X_INSERTED: Here VBA enters a 1 after the double-clicked row has been transferred to the target.

Target pointer

The following pointers are mandatory:

- SOURCE: Fields whose contents are taken from the source.
- ROW: The x marks the row below which the data range begins.
- KEY: sequential row number generated by VBA.

Optional pointer

- SORT: Sort after expanding the table.

  The digits 11, 21 ,31 ... indicate which characteristics are sorted in which order DESCENDING.

  The digits 12, 22, 32 ... indicate which characteristics are sorted in which order ASCENDING.

Optional pointers for variants

- INTERMEDIATE: Field contents to be taken from the Intermediate.
- FILL: here you can specify in how many rows of the target the source values are entered. FILL is only possible in connection with the dialog module.

16 Expand

Expand is a corner that basically multiplies 2 tables (Source and Intermediate) with each other: If the table in the ExpandXXSource corner contains 10 rows and the table in the ExpandXXIntermediate corner contains 5 rows, 50 rows are generated in the table in the "ExpandXXTarget" corner.

Example: the customer wants to plan 5 personnel cost types (Intermediate) for 10 employee master data (source). Expand creates a table with 10 x 5 = 50 rows, 1 row for each employee-cost element combination.

Another application is, for example, the valuation of a table with planned sales data (source) with production cost data (intermediate) with material price, material overhead price and production cost price. In the first line, the sales quantity must be multiplied by the material price, in the second line the sales quantity by the material overhead price and so on.

It should be noted here that the planned sales data in the source does not only contain one material, but several, and the production costs in the intermediate do not only contain the production costs of one material. The RULE pointer can be used to compare one or more key values in the source table with the corresponding key values in the intermediate table and only the intermediate rows for which these are identical, in this example the material number, can be transferred for each source row.

16.1 Brief overview

Area	Characteristic	Info
Code		
VBA	basKitExpand	
	clsKitExpand	
	clsKitExpandSource	
	clsKitExpandIntermediate	
	clsKitExpandSource	
Settings		
Configuration sheet	Expand	
Settings table	ExpandAreas	
Controls		
Macro	ExpandActivate	
Impact Area		
Corner	ExpandXXSource	
	ExpandXXIntermediate	
	ExpandXXTarget	

16.2 Settings

The desired expand corners are activated via the [ExpandAreas] table:

ExpandAreas

Id	Active
Expand01	1

The ExpandActivate macro is used to execute all ExpandCorner of the table. The macro must be entered in a UserExit.

NOTE: If you want to execute individual corners at different times (e.g. AFT_LEAD_SAT, AFT_READ), this must be added on a project-specific basis.

New in version 4.3: Expand has been extended so that it also works in MultiPage. To do this, however, the source, intermediate and target must be on the same sheet

Set up the Source

The following pointers are mandatory:

- TARGET: Marks the columns in which the fields to be transferred to the target must be entered.
- ROW: Marks the row below which the rows have to be transferred with an X.
- X_INSERT: Controls whether the row is transferred.
- X_INSERTED: Is updated after execution.

This pointer is optional:

- INTERMEDIATE: Marks the columns that are to be used for filtering the intermediate.

Expand01Source

	ROW	X_INSERT	X_INSERTED					
TARGET					ROW_S	SOURCE1	SOURCE2	INT1
INTERMEDIATE								INT1
	X	2			ROW	SOURCE1	SOURCE2	INT1
		2	1		1	A	E	2020

Set up the Intermediate

NOTE: The intermediate table must be created as a **structured table**.

The following pointers are mandatory:

- TARGET: Marks the columns in which the fields to be transferred to the target must be entered.
- ROW: Marks the row below which the rows must be transferred with an X.

These pointers are optional:

- SOURCE: Marks the columns that are used to filter the rows to be transferred from the intermediate to the target.
- RULE: The filter rule. Possible entries: = | < | > | <> | <= | >=
- You can also enter * in the lines of the Intermediate, in which case the filter rule is ignored for this line and the entry is still applied.
- The filter value HAS to be transferred to the target for technical reasons, but the value does not have to be transferred there.
- Example: Line 1 is only transferred for the line from the target if the AssetClass in the source and intermediate are identical.

Expand01Intermediate

	ROW				
TARGET			ROW_I	INT1	INT2
SOURCE				INT1	
RULE				>	
	X		ROW	INT1	INT2
			1	2022	FC

Set up the Target

NOTE: The target table must be created as a **structured table**.

The following pointers are mandatory:

- SOURCE: Marks the columns into which the fields from the source are transferred.
- INTERMEDIATE: Marks the columns into which the fields from the intermediate are transferred.
- ROW: Marks with X the row below which the rows from the source and intermediate are created.
- KEY: The index of the row generated by the expand.

Expand01Target

	ROW			KEY				
SOURCE					ROW_S	SOURCE1	SOURCE2	
INTERMEDIATE								ROW_I
	X			Row	ROW_S	SOURCE1	SOURCE2	ROW_I
				1	1	A	E	1
				2	1	A	E	2

16.3 Definitions and notes

Usage of formulas in ExpandTarget

Formulas are deleted from the structured table for performance reasons.

A FormulaCorner can then be executed via the UserExit AFT_EXPAND, see chapter 12. to generate the formulas.

ColumnPointer "ROW"

A trunk corner is used for the ExpandTarget, which is why the ColumnPointer "ROW" must be used in the FormulaCorner instead of the ColumnPointer "ROWALL".

Usage of X_INSERT

X_INSERT is used to control whether and how the lines are transferred from the target:

0 = no transfer

1 = one-time transfer

2 = repeated transfer

If a line is only to be transferred once, X_INSERT must also be present as a field in the ExpandTarget and be transferred from the ExpandSource.

If X_INSERT is missing in the ExpandTarget, these entries are deleted when a new additional line is created and Expand is executed again.

Header colors in the Expand tables

For better readability, the header columns of the ExpandTarget are to be provided with the following background/font colors. The colors are also to be used for the headers of the tables themselves.

Origin	R	G	B	R	G	B
Source	0	99	172	255	255	255
Intermediate	255	179	0	0	0	0

This gives a quicker overview of the data origin of the individual columns in the ExpandTarget.

17 Flip

Flip01

	FLIPSUM	FLIPFLAG	ROW	FLIPLEVEL					

The Flip module offers the possibility to expand the subordinate hierarchy levels of a cost element structure with a double click and to hide them again.

Brief overview

Area	Feature	Info
Code		
VBA	basKitFlip	
	clsKitFlip	
Settings		
Configuration sheet	Flip	
Setting table	FlipAreas	
	FlipConfig	
Controls		
Macro	InitFlipCorner	
UserExits	InitFlipCorner	
Impact		
Corner	FlipXX	
KeyPointer	FLIPSUM	A flag for control, is set by code
	FLIPFLAG	Entry of formulas that set a flag ("X") for the initial view
	FLIPLEVEL	Indicates the column with the details of the totals hierarchy structure for double-click collapsing and expanding
	ROW	Below the entry "X" the plan area begins.

17.1 Settings

Functionality

The flip module only works in cooperation with the navigation!

In the navigation, the flags ("1"), which are set by macro in the expander corner, must be evaluated by formulas in order to enable fade-in and fade-out by double-click.

Initial view

For the desired initial output view in the data sheet, the UserExit macro *InitFlipCorner* must be executed initially (possible UserExit events AFT_READ, OPEN_IN_SAP, WB_OPEN).

In the first step, the *InitFlipCorner* macro sets the flag in the customizing table [ZZCustomizingFlagsBase] to *true* and then evaluates the flags set by formula (see example figure below and also point 1: Set up MOD Expander 3.5).

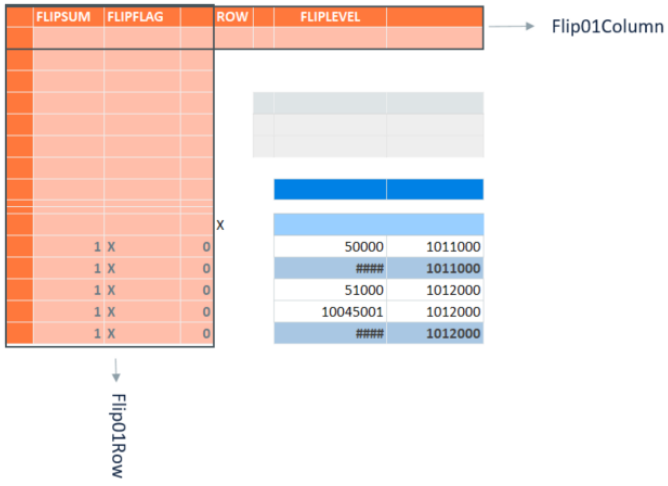
Per macro, a "1" is now placed in the rows of the EXPANDSUM-KeyPointer column where a flag (e.g. "X") is present in the rows of the EXPANDPARAM-KeyPointer column.

Subsequently, the flag in the customizing table [ZZCustomizingFlagsBase] is set to *false* again, so that the formula (see example below) sets an "X" in each row.

If the navigation view is now executed as usual (e.g. AFT_READ), all rows with "1" are displayed (provided, of course, that the "1" is evaluated in the navigation formula).

Set up corner

The corner is set up as usual: {FlipXyColumn} and {FlipXyRow}.



Corner Flip01

Configuration tables

The desired FlipCorner are activated via the [FlipAreas] table:

FlipAreas

Id	Activate
Flip01	1
Flip02	0

FlipAreas table

The configuration columns *InitialFlag* and *SumFlag* are maintained in the [FlipConfig] table:

FlipConfig

Id	InitialFlag	SumFlag
Flip01	0	*
Flip02	0	#

FlipConfig table

The execution of the initial macro for a specific corner is controlled by the additional specification of the corner in the [Customizing] table.

Customizing

Process	SubProcess	Parameter	Key	Description	Value
UserExit	Macro	OPEN_IN_SAP			
UserExit	View	AFT_READ			
UserExit	Macro	AFT_READ			
UserExit	Macro	WB_OPEN			
UserExit	Macro	WB_OPEN			InitFlipCorner;Flip01
UserExit	Macro	AFT_LEAD_SAT			
UserExit	Macro	AFT_LEAD_SAT			
UserExit	Macro	AFT_PLAN			

Customizing table

KeyPointer

FLIPSUM

A "1" is placed in this column by code; initially via macro *InitFlipCorner*, if a flag ("X") is set in the same row of the EXPANDPARAM column, or on double-click depending on the summation hierarchy level.

FLIPFLAG

Entry of formulas that set a flag ("X") for the initial view.

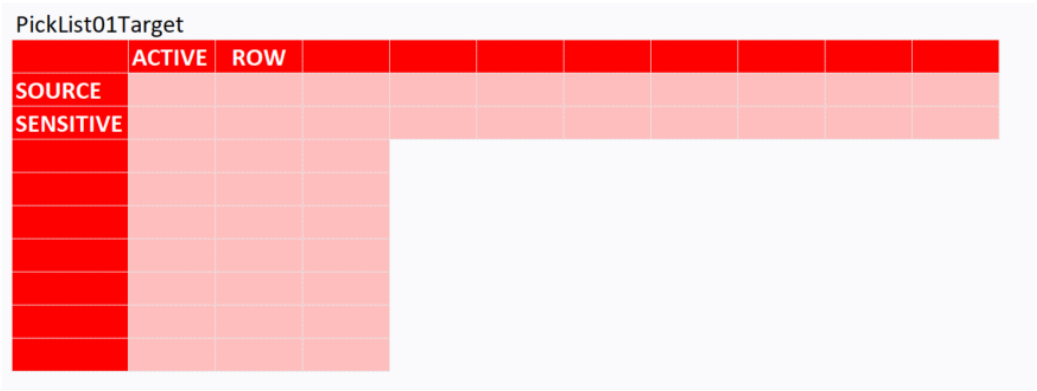
FLIPLEVEL

Indicates the column with the details of the totals hierarchy structure for double-click collapsing and expanding

ROW

Below the entry "X" the plan area begins.

18 PickList



The PickList is able to pick up data from one table and transfer it to another table.

Brief overview

Area	Feature	Info
Code		
VBA	basKitPickList	
	frmKitPickList	
	clsKitPickList	
	clsKitPickListSource	
	clsKitPickListTarget	
Controls		
Configuration sheet	PickList	
Setting table	PickListAreas	
Settings		
Macro	PickListActivate	
Cado		
UserExits	ATF_PICKLIST	
Impact Area		
Corner	PickListXXSource PickListXXTarget	
KeyPointer Source	TARGET	Fields whose contents are copied from the target to the dialog.
	ROW	"X" marks the row from which the data range starts.
	WIDTH	(optional) Column width in dialog box
KeyPointer Target	SOURCE	Fields whose contents are to be transferred from the source or the dialog to the target.

	SENSITIVE	Fields to which the PickList should react via double-click or shortcut
	ACTIVATE	(optional) Marks all rows to which the PickList should react.
	ROW	"X" marks the row of which the data range starts.

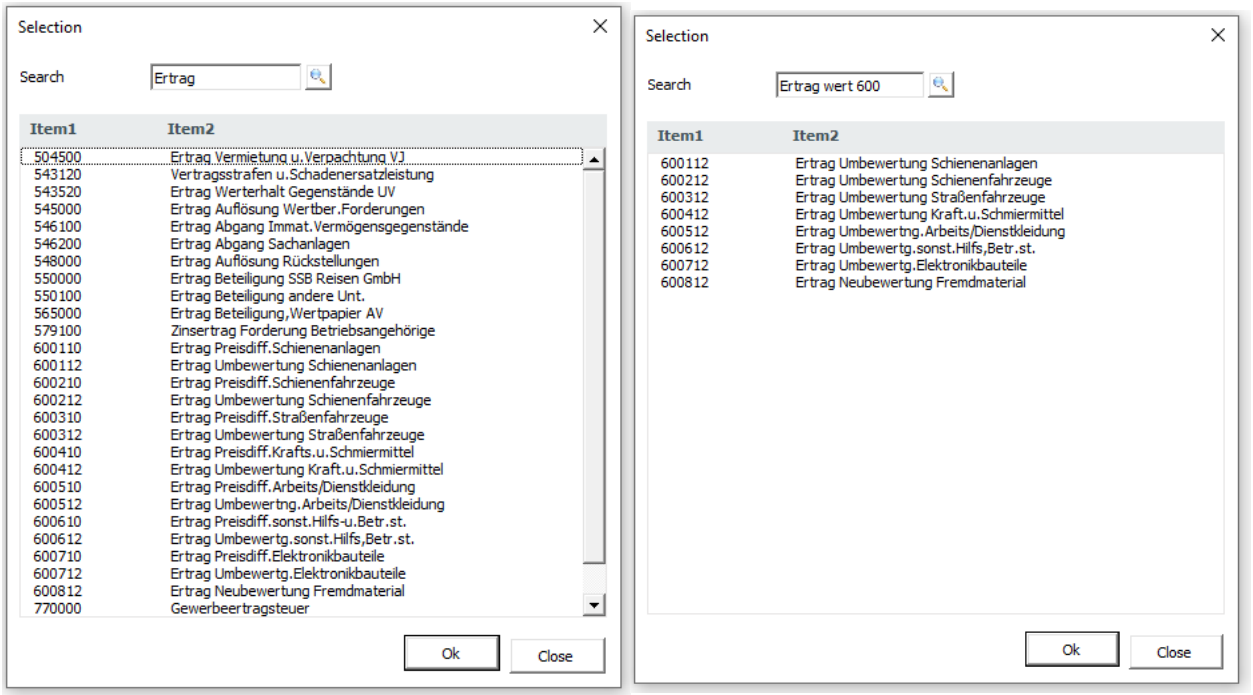
18.1 How the PickList works

- If the cursor is positioned in one of the SENSITVE columns of the Target-Corer, the dialog can be displayed by pressing the key specified in the settings (in our example F4) or by double-clicking. In contrast to the old PickList, the cell protection is ignored.
- If an entry is selected with a double-click or Enter, the values are entered in the active row. The entries in the target corner at Pointer SOURCE determine which column values are transferred from the user form or the source. The order of the entries is freely selectable.
- The search in the search field goes through all columns and finds substrings.
- With the Close button or Esc you can leave the dialog without inserting anything.

Search in the Dialog

The contents, column headings and widths of the dialog are set up in the source.

The search field can be used to search for specific source entries. All lines containing the search term are found (partial terms/substrings are also possible). Upper and lower case is not important. The search option has been extended so that several terms - separated by a blank - can be entered. The hit list then contains those lines for which all search terms apply (AND link).



18.2 Controls

The PickListActivate macro cannot be called via UserExits or similar. It is always activated either by a shortcut or by double-click.

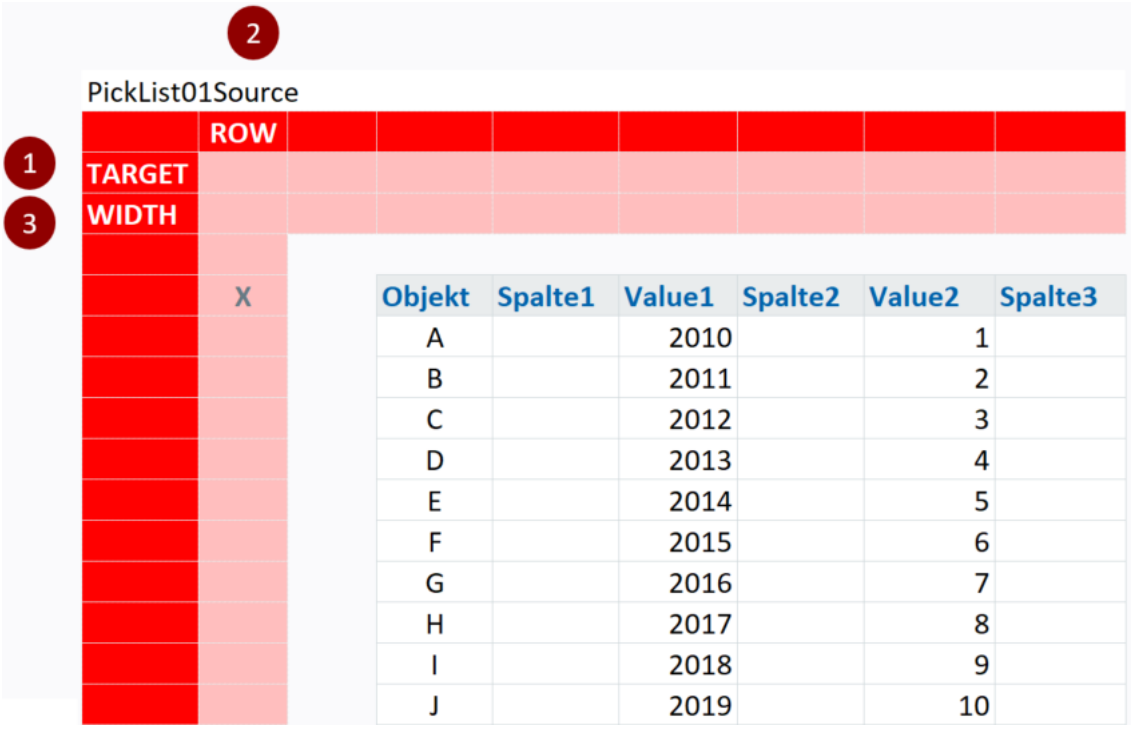
To call up the PickList with a shortcut, a corresponding entry is made in the Settings on the Customizing sheet.

18.3 Settings

Set up source

For the source corner, the PickListXySourceColumn or PickListXySourceRow areas are set up.

As data area a structured table is always expected, therefore the source corner is a *trunk corner* (like e.g. with the satellite All).



Objekt	Spalte1	Value1	Spalte2	Value2	Spalte3
A		2010		1	
B		2011		2	
C		2012		3	
D		2013		4	
E		2014		5	
F		2015		6	
G		2016		7	
H		2017		8	
I		2018		9	
J		2019		10	

Corner PickList01Source

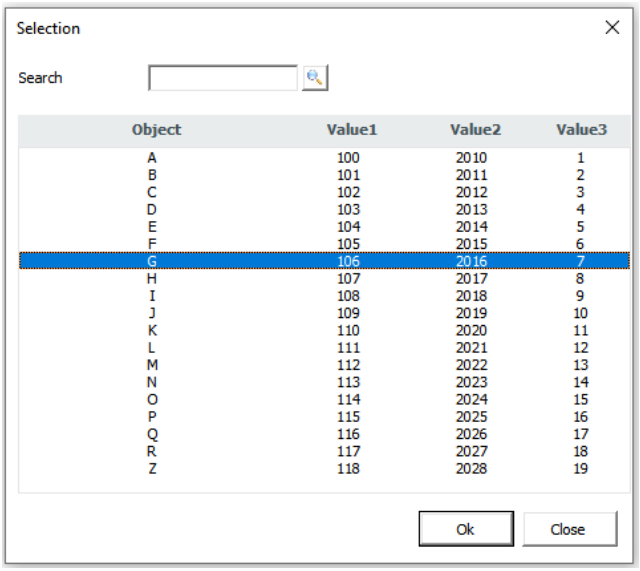
TARGET:

Fields whose contents are copied from the target to the dialog.

ROW:

The x marks the row below which the data range begins.

The header identifiers and formats are taken from the table. The result will look like this:



WIDTH: Column width (optional)

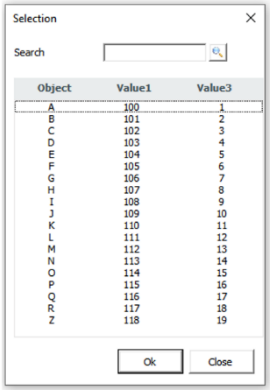
The column width in the dialog is usually taken from the column widths in the source table. In at least two cases, however, this is unfavorable:

- a data column is unintentionally hidden (e.g. by navigation). Consequence: the data column is not displayed in the dialog because the width is equal to 0
- a data column is to be transferred to the target, but not displayed in the dialog. However, hiding the column is not desired.

The WIDTH pointer is suitable for these cases. It can be used to override the reading of the column width from the table for the dialog.

PickList01Source

	ROW								
TARGET			Item1			Item2		Item3	
WIDTH				10		0			
	X		Objekt	Spalte1	Value1	Spalte2	Value2	Spalte3	
			A		2010		1		
			B		2011		2		
			C		2012		3		
			D		2013		4		
			E		2014		5		
			F		2015		6		
			G		2016		7		
			H		2017		8		



Use one source for different targets

You can place several source corners over a source table. The respective target corners can then be located at different places in the master.

PickList01Source

	ROW								
TARGET			Item1					Item4	
		PickList01Source							
		ROW							
		TARGET		Item1		Item2	Item3	Item4	
		WIDTH			10		0		
			X	Objekt	Spalte1	Value1	Value2	Value3	Spalte3
				A		100	2010	1	
				B		101	2011	2	
				C		102	2012	3	
				D		103	2013	4	
				E		104	2014	5	
				F		105	2015	6	
				G		106	2016	7	
				H		107	2017	8	

Set up target

The {PickListXyTargetColumn} or {PickListXyTargetRow} areas are set up for the target corner.

Pointer

PickList01Target Corner

SOURCE:

Fields whose contents are to be transferred from the source or the dialog to the target.

SENSITIVE:

One or more fields to which the PickList should react by double-click or shortcut (everything except *empty* allowed, here: X).

ROW:

The x marks the row below which the data range begins.

ACTIVE (optional)

With the optional pointer ACTIVE you can mark all rows with "1" to which the PickList should react. This can be used to exclude rows that are not to be filled via PickList (e.g. totals rows).

Nesting multiple PickList

If several PickList target corners are superimposed, the pointers for SENSITIVE must not be in the same column, otherwise VBA cannot clearly determine which corner must be served.

Corner PickList02Target and PickList01Target

Setting tables

PickList Areas

The desired PickList corners are activated via the [PickListAreas] table:

PickListAreas

Id	Activate
PickList01	1

PickListAreas table

Customizing:

Here the shortcut for calling the PickList is defined.

Calling by double-click is always possible, regardless of whether a shortcut has been stored or not.

Customizing

Process	SubProcess	Parameter	Key	Description	Value
Settings		PickListKey		Select a shortcut key	F4

Customizing table

19 Placier

In Allevo it is possible to read the cost element structure only with the posted cost elements. This is useful if many objects are to be read. Placier now offers the option of adding cost elements that have not yet been posted to via a selection dialog in the hierarchy.

In 4.3, Placier can be used to insert lines for missing object-element combinations when working with the dynamic corner. Only rows with SumLevel=0 are possible.

Brief overview

Area	Characteristic	Info
Code		
VBA	basKitPlacier	
	frmKitPlacier	
	clsKitPlacier	
	clsKitPlacierInsert	
	clsKitPlacierNode	
	clsKitPlacierSource	
	clsKitPlacierTarget	
Controls		
Configuration sheet	Placier	
Settings table	PlacierAreas	
Settings		
Macro	PlacierStart	The PlacierStart function is called up via the ribbon
Cado	CadoAfterPlacier	
UserExits	PLACIER_AFT_INSERT_ROW, PLACIER_AFT_EDIT_ROW	optional

19.1 Set up the Placier

The desired Placier Corners are activated via the PlacierAreas table:

PlacierAreas

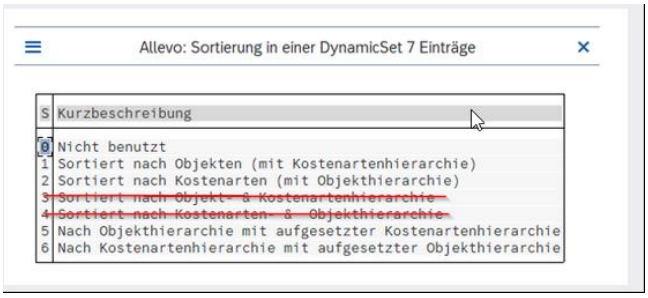
Id	Active
Placier01	1

The PlacierStart macro is usually called up via the menu ribbon.

19.2 Settings in Allevo

There are a few things to consider on the ABAP side

Selection DynamicSet



Select Preselect

Deactivate auxiliary summation levels

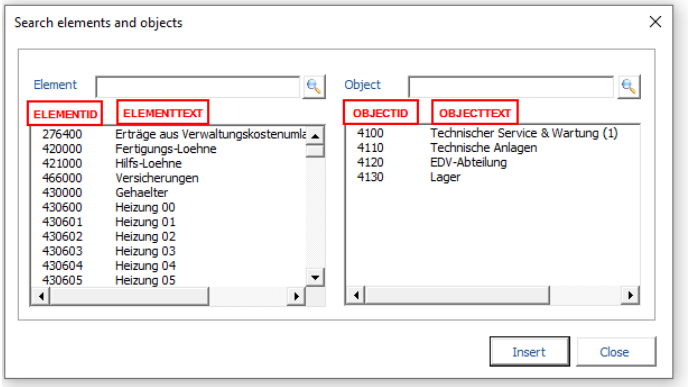
19.3 Set up Corner

Source:

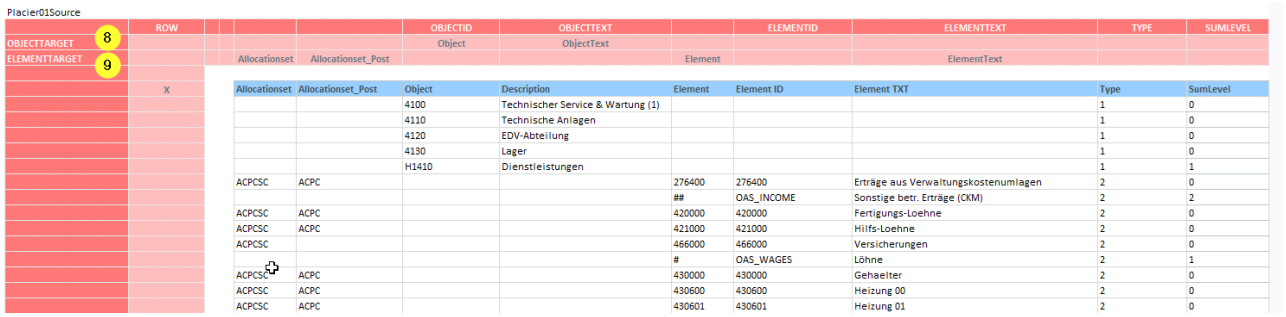
Placier01Source

	ROW 1			OBJECTID 2	OBJECTTEXT 3		ELEMENTID 4	ELEMENTTEXT 5	TYPE 6	SUMLEVEL 7
OBJECTTARGET				Object	ObjectText					
ELEMENTTARGET		Allocationset	Allocationset_Post			Element		ElementText		
	x	Allocationset	Allocationset_Post	Object	Description	Element	Element ID	Element TXT	Type	SumLevel
				4100	Technischer Service & Wartung (1)				1	0
				4110	Technische Anlagen				1	0
				4120	EDV-Abteilung				1	0
				4130	Lager				1	0
				H1410	Dienstleistungen				1	1
		ACPCSC	ACPC			276400	276400	Erträge aus Verwaltungskostenumlagen	2	0
						##	OAS_INCOME	Sonstige betr. Erträge (CKM)	2	2
		ACPCSC	ACPC			420000	420000	Fertigungs-Loehne	2	0
		ACPCSC	ACPC			421000	421000	Hilfs-Loehne	2	0
		ACPCSC				466000	466000	Versicherungen	2	0
						#	OAS_WAGES	Löhne	2	1
		ACPCSC	ACPC			430000	430000	Gehaelter	2	0
		ACPCSC	ACPC			430600	430600	Heizung 00	2	0
		ACPCSC	ACPC			430601	430601	Heizung 01	2	0
		ACPCSC	ACPC			430602	430602	Heizung 02	2	0
		ACPCSC	ACPC			430603	430603	Heizung 03	2	0

1. or ROW, the x marks the row below which the data range begins.
2. to 5. The column contents that are displayed in the UserForm must be marked with OBJECTID, OBJECTTEXT, ELEMENTID and ELEMENTTEXT:



6. TYPE specifies whether it is an object (=1) or an element (=2).
7. SUMLEVEL defines the sum level.

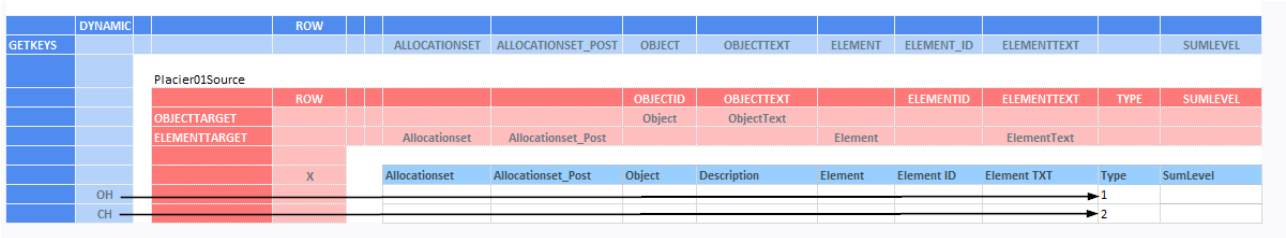


8. - 9. The OBJECTTARGET and ELEMENTTARGET pointers are optional. Columns for objects or elements with freely selectable identifiers can be identified there. The contents of these columns are transferred to the target. It should be noted that the information for objects and elements in the target is in one line (pointer SOURCE). The identifiers must therefore be unique across all items, i.e. you cannot just use "Text" instead of ObjectText and ElementText both times.

Best Practice

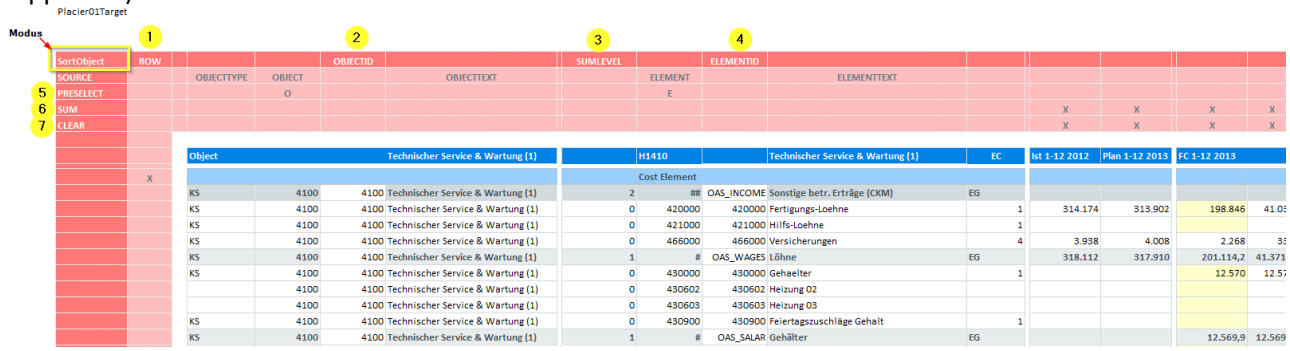
The source is usually filled via the DynamicCorner. The GETKEYS pointers OBJECT, OBJECTTEXT, ELEMENT_ID, ELEMENTTEXT and SUMLEVEL are mandatory. The ELEMENT, ALLOCATIONSET and ALLOCATIONSET_POST pointers for transfer to the target should also be useful in most cases.

The entries 1 (=object) and 2 (=element) can be hard-coded for the TYPE placement pointer if the entries OH (=object) and CH (=element) are available for DYNAMIC as shown in the image below. The dynamic corner also copies the entries. If you prefer, you can also operate with formulas.

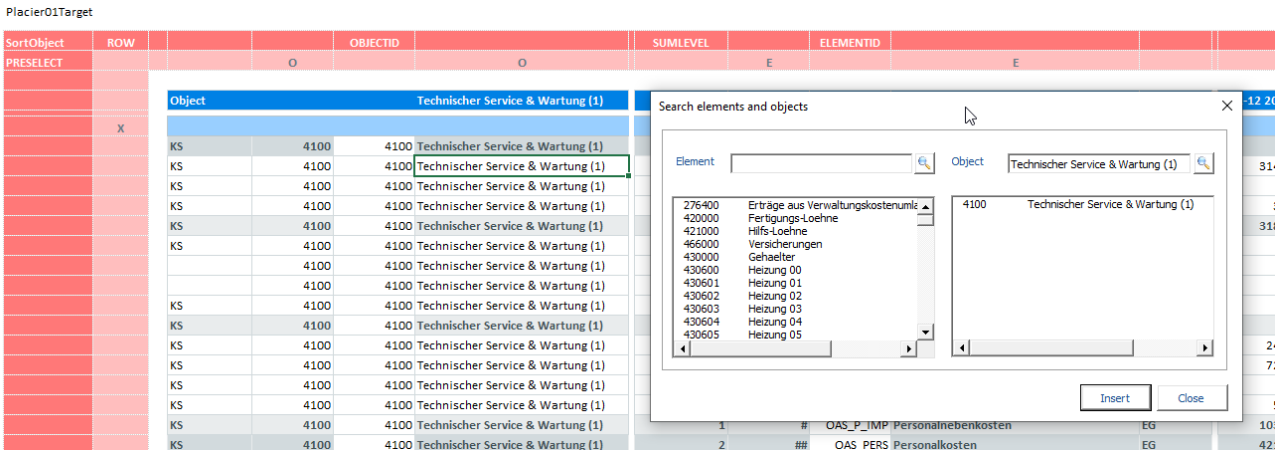


Target

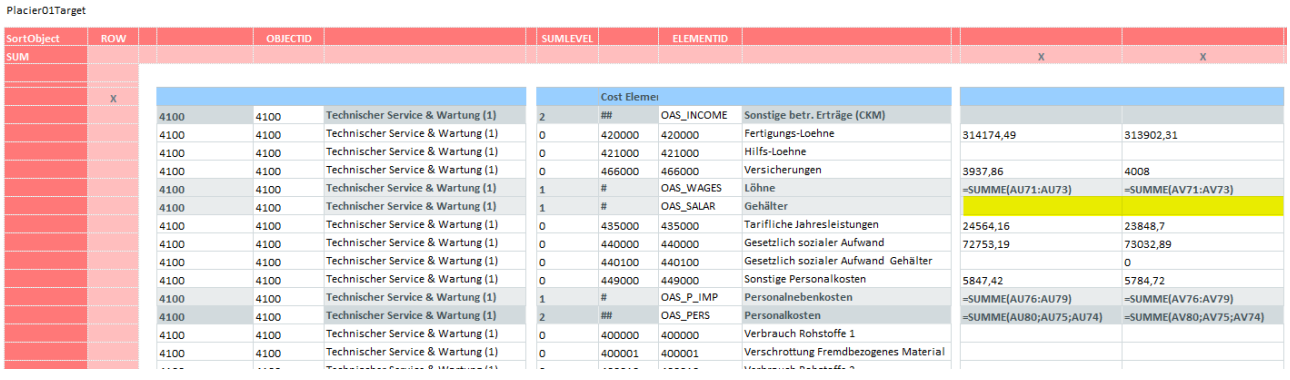
The sort mode is specified in the cross corner of the corner. The mode is either SORTOBJECT or SORTELEMENT. SORTOBJECT corresponds to DynamicSet 1 and 5, SORTELEMENT to 2 and 6 (variants 3 and 4 are not supported).



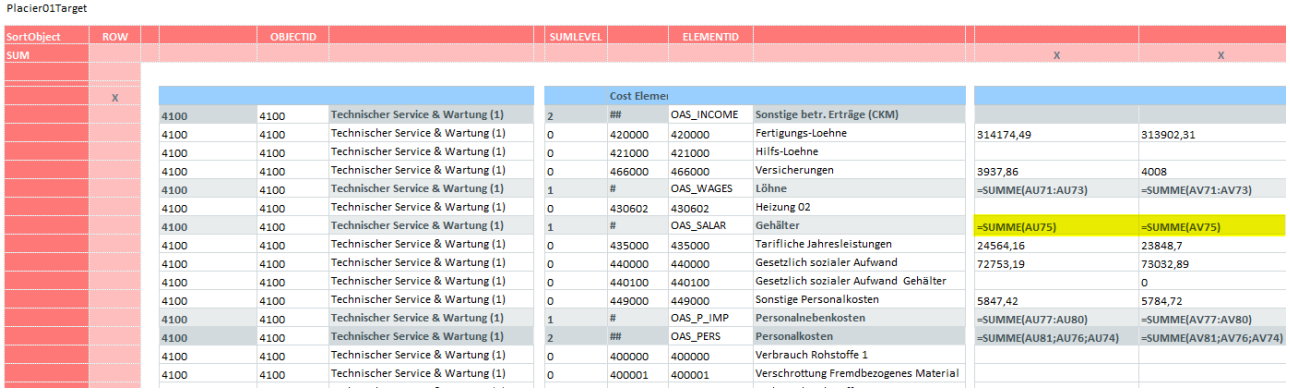
1. in ROW, the x marks the line below which the data range begins.
2. OBJECTID: unique ID, which must also be present in the source.
3. SUMLEVEL: summation level that must match the one in the source.
4. ELEMENTID: unique ID, which must also be present in the source.
5. PRESELECT. The entries "O" and "E" mark the columns whose selection triggers the preselection of the UserForm. If the cursor is positioned on a target column that was marked with "O" or "E" via the PRESELECT pointer, the respective object/element is preselected in the dialog (except in totals rows). If the cursor is positioned on an unmarked column, all items are listed.



6. in SUM, the columns are marked with "x" for which Placier may have to add a summation formula after inserting rows.



Before



After

7. with CLEAR, the columns whose contents are to be deleted from the new row after copying/pasting the row are marked with "x".

PlacierO1Target

SortObj	ROW		OBJECTID	SUMLEVEL	ELEMENTID								
CLEAR							X	X	X	X	X	X	X
	X			Cost Element									
		4100	4100	Technischer Service & Wartung (1)	2 ##	OAS_INCOME	Sonstige betr. Erträge (CKM)						
		4100	4100	Technischer Service & Wartung (1)	0 420000	420000	Fertigungs-Loehne	314.174	313.902	198.846	41.035	27.378	25.701	26.074
		4100	4100	Technischer Service & Wartung (1)	0 421000	421000	Hilfs-Loehne							
		4100	4100	Technischer Service & Wartung (1)	0 466000	466000	Versicherungen	3.938	4.008	2.268	337	341	317	311
		4100	4100	Technischer Service & Wartung (1)	1 #	OAS_WAGES	Löhne	318.112	317.910	201.114,2	41.371,9	27.718,4	26.018,4	26.384,7
		4100	4100	Technischer Service & Wartung (1)	0 430602	430602	Heizung 02							
		4100	4100	Technischer Service & Wartung (1)	1 #	OAS_SALAR	Gehälter							
		4100	4100	Technischer Service & Wartung (1)	0 435000	435000	Tarifliche Jahresleistungen	24.564	23.849	23.849				
		4100	4100	Technischer Service & Wartung (1)	0 440000	440000	Gesetzlich sozialer Aufwand	72.753	73.033	76.057	8.884	5.548	5.708	5.385
		4100	4100	Technischer Service & Wartung (1)	0 440100	440100	Gesetzlich sozialer Aufwand Gehälter			2.443	2.443			
		4100	4100	Technischer Service & Wartung (1)	0 449000	449000	Sonstige Personalkosten	5.847	5.785	6.310	955	508	475	504
		4100	4100	Technischer Service & Wartung (1)	1 #	OAS_P_IMP	Personalnebenkosten	103.165	102.666	108.659,1	12.282,1	6.056,2	6.182,6	5.888,2
		4100	4100	Technischer Service & Wartung (1)	2 ##	OAS_PERS	Personalkosten	421.277	420.577	309.773,3	53.654,0	33.774,7	32.201,0	32.272,9
		4100	4100	Technischer Service & Wartung (1)	0 400000	400000	Verbrauch Rohstoffe 1			24.342				
		4100	4100	Technischer Service & Wartung (1)	0 400001	400001	Verschrottung Fremdbezogenes Material			6.666				

Optional:

PlacierO1Target

SortObject	ROW							OBJECTID		SUMLEVEL		ELEMENTID	
SOURCE					ALLOCATIONSET	ALLOCATIONSET_POST	OBJECTTYPE	OBJECT	OBJECTTEXT		ELEMENT		ELEMENTTEXT
PRESELECT								O			E		
SUM													
CLEAR													

Dynamic01

DYNAMIC														
GETKEYS					ALLOCATIONSET	ALLOCATIONSET_POST	OBJECTTYPE	OBJECT	OBJECT_ID	OBJECTTEXT	SUMLEVEL	ELEMENT	ELEMENT_ID	ELEMENTTEXT

Dialog01

READ	ALLOCATIONSET				OBJECTTYPE	OBJECT						ELEMENT		
WRITE			ALLOCATIONSET_POST		OBJECTTYPE	OBJECT						ELEMENT		
	X													
		CE												
		CE			ACPCSC	ACPC								
		CE			ACPCSC	ACPC								
		CE			ACPCSC									
		CE												
		CE			ACPCSC	ACPC								
		CE			ACPCSC	ACPC								
		CE			ACPCSC	ACPC								
		CE			ACPCSC	ACPC								
		CE												
		CE			ACPCSC	ACPC								
		CE			ACPCSC	ACPC								

Object			Technischer Service & Wartung (1)	H1410			Technischer Service & Wartung (1)
				Cost Element			
KS	4100	4100	Technischer Service & Wartung (1)	2	##	OAS_INCOME	Sonstige betr. Erträge (CKM)
KS	4100	4100	Technischer Service & Wartung (1)	0	420000	420000	Fertigungs-Loehne
KS	4100	4100	Technischer Service & Wartung (1)	0	421000	421000	Hilfs-Loehne
KS	4100	4100	Technischer Service & Wartung (1)	0	466000	466000	Versicherungen
KS	4100	4100	Technischer Service & Wartung (1)	1	#	OAS_WAGES	Löhne
KS	4100	4100	Technischer Service & Wartung (1)	0	430000	430000	Gehaelter
	4100	4100	Technischer Service & Wartung (1)	0	430602	430602	Heizung 02
	4100	4100	Technischer Service & Wartung (1)	0	430603	430603	Heizung 03
KS	4100	4100	Technischer Service & Wartung (1)	0	430900	430900	Feiertagszuschläge Gehalt
KS	4100	4100	Technischer Service & Wartung (1)	1	#	OAS_SALAR	Gehälter
KS	4100	4100	Technischer Service & Wartung (1)	0	435000	435000	Tarifliche Jahresleistungen
KS	4100	4100	Technischer Service & Wartung (1)	0	440000	440000	Gesetzlich sozialer Aufwand

8. with CLEAR, the columns whose contents are to be deleted from the new row after copying/pasting the row are marked with "x".

19.4 How Placier works

Placier is usually called via the ribbon, for this purpose a CustomButton with macro PlacierStart is set up on the sheet on which the target corner is located.

When PlacierStart is called up, a UserForm opens in which a combination of both can be selected from an object list and an element list. The displayed objects and elements are read from the source.

The following happens when the dialog is opened:

If the cursor is positioned on a target column that has been marked with "O" or "E" via the PRESELECT pointer, the respective object/element is preselected in the dialog (except in totals rows). If the cursor is positioned on an unmarked column, all items are listed.

The search fields Element and Object can also be used to search for partial strings. If the search field is empty, the entire list is displayed.

If the Insert button is pressed, the system checks whether the selected combination already exists. The following then happens:

Case 1: **The combination already exists.**

A corresponding message is displayed. The user can decide whether to cancel or jump to the line. In the latter case, the line is displayed and activated.

If a caller or a macro is stored for UserExit PLACIER_AFT_EDIT_ROW, it is executed.

Case 2: **The combination is not yet included in the planning screen.**

Then

the position for a new line in the target is searched for according to the sequence of OBJECTID and ELEMENTID in the source

a target line with all values/formatting is copied and pasted at the determined position. (See also Best Practice)

In the parent item, the summation formula is inserted or adjusted in all columns that are marked with "x" in the target at Pointer SUM. (See also Best Practice)

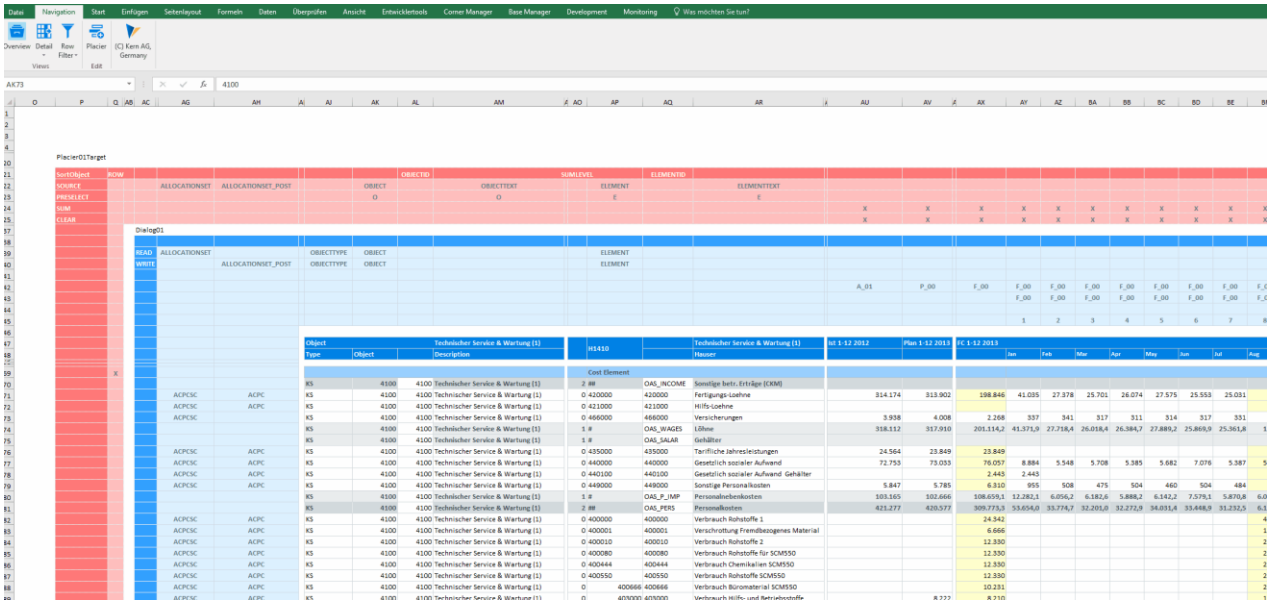
depending on the setting, copied values are deleted in the new line. (The cells to be deleted are determined via the CLEAR pointer in the target corner).

the IDs and the sum level are entered in the new line (pointers OBJECTID, ELEMENTID and SUMLEVEL)

the values/texts from the source (pointers OBJECTTARGET and ELEMENTTARGET) are entered in the target line (pointer SOURCE)

the callers and macros stored in UserExit PLACIER_AFT_INSERT_ROW are executed

the new line is selected and activated



Best Practice

In the dynamic structure of the planning screen, the **totals formulas** are usually **generated** once after reading (from 4.3 with the FormulaCorner). However, it is NOT necessary to call up the totals generator again after Placier has inserted a line, because Placier determines and inserts the totals formula for the respective parent item itself.

Placier does not insert empty lines, but copies a line and inserts it with all formatting in the correct position. The formatting is probably 99% correct. If they do not fit, you can use the inline variant of the StyleCorner (from 4.3) with UserExit **PLACIER_AFT_INSERT_ROW**.

19.5 CadoAfterPlacier

To call a macro after inserting the new line, the UserExit PLACIER_AFT_INSERT_ROW will suffice in most cases. If the inserted object/element is to be accessed, additional customized functions can be programmed with CadoAfterPlacier.

20 PrintView

Brief overview

Area	Feature	Info
Code		
VBA	basKitPrintView	
	clsKitPrintView	
	clsKitPrintViewSetup	
Settings		
Configuration sheet	PrintView	
Setting table	PrintViewCommonSettings	
	PrintViewBase	
Controls		
Macro	PrintView	
	PrintMultipleViews	
Cado	CadoPrintViewSetup	
Impact Area		

20.1 Controls

Functionality

Basically, the print function uses the page settings in the master. This concerns scaling, margins, footer and header etc.. Only the print area and the row repetition must be determined by VBA.

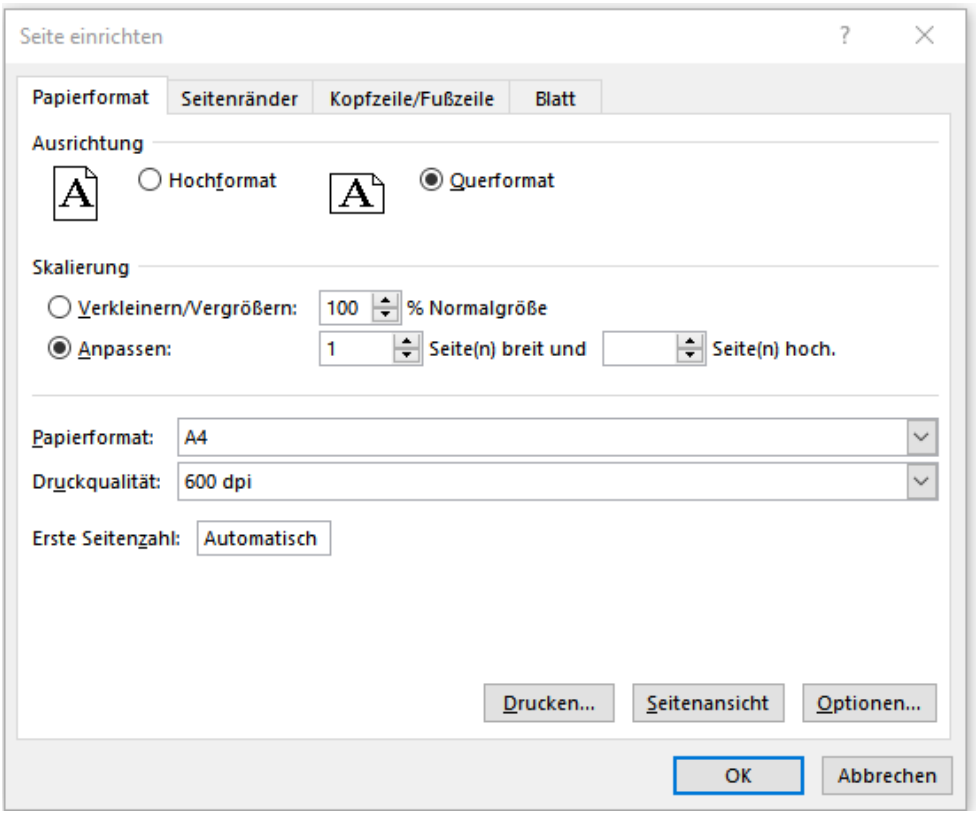
To determine the print area, a navigation corner is always necessary, which includes all rows to be printed. The row repetition is determined by VBA using the freeze pointer. If no freeze is set up, no row repetition takes place.

Two functions are available to print navigation views:

- The **PrintView** function prints the active navigation view on the active sheet
- The **PrintMultipleViews** function prints all stored navigation views in the specified order

PrintView - Page Setup

Basically, the print function uses the page settings in the master. The following settings should be sufficient as a rule:



However, these settings then apply to all views on this worksheet. Sometimes it is necessary to specify different settings for the views of a worksheet. This can be done via the table [PrintViewBase] - both for the single print and for PrintMultipleViews.

PrintViewBase

SheetName	View	Orientation	PaperSize	Zoom	FitToPage	Order
Allevo	Allevo01	L		30		
Allevo	Allevo02	P		100		
Allevo	Allevo03	L			1|0	

1 2 3 4 5

View — Skalierung

1. View

Sheet name and view id are used to uniquely identify the view.

2. Orientation

The orientation of the page can be specified with L (=landscape) or P (=portrait).

3. PaperSize

For the paper size, the defaults can be A3, A4 or A5. If you need more exotic sizes, you can enter the numeric value from this list: Paper size

4. Scaling

The scaling can be done either via the zoom or the page adjustment. Both at the same time is not possible.

- Zoom: values between 10 and 400
- Page Fit: In the column *FitToPage the* first value stands for the number of "Pages wide" and the second for "Pages high". The values are separated with "|".

5th order

For the function PrintMultipleViews the order of the views to be printed must be entered in the column Order. The points 1-4 are also valid for this function, if something is entered. If not, the Excel page settings apply.

PrintViewBase

SheetName	View	Orientation	PaperSize	Zoom	FitToPage	Order
Allevo	Allevo01					3
Allevo	Allevo02					1
Allevo	Allevo03					2

Note:	**Multipage** In Multipage, the views of a template are printed sorted sheet by sheet, i.e. all views of the first sheet, then those of the second, etc.

Set up additional functions

The following extras are available for the print function

- Show dialogs before printing
- Preset page settings for views
- Print multiple views in specified order

1. If the PrintMultipleViews macro is called instead of the PrintView macro, the order of the views to be printed must be specified.
2. What unfortunately doesn't work: generate a contiguous pdf with PrintMultipleViews, because each view has to be sent to the printer individually.

Show dialogs before printing

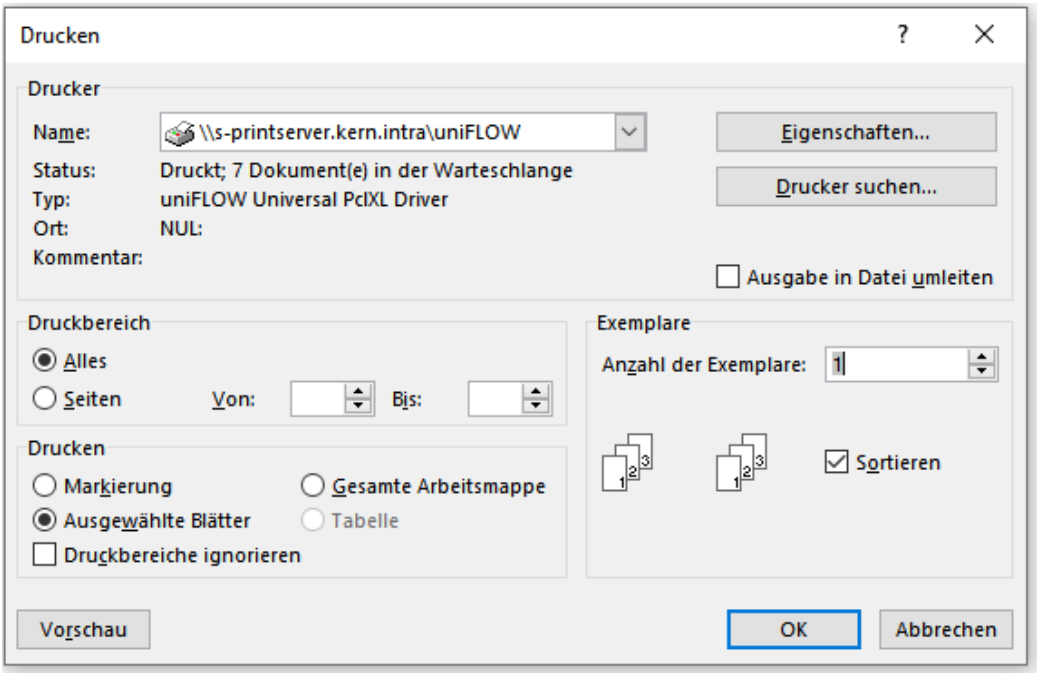
The [PrintViewCommonSettings] table can be used to control whether and which dialog is to be displayed before printing. Value "1" activates the respective dialog.

PrintViewCommonSettings

	Id	Value
1	ShowPrinterDialog	0
2	ShowPageSetupDialog	0
3	ShowPrintViewsWarning	0

1. showPrinterDialog

With the print dialog you can select a printer, but the print area is determined by VBA and cannot be changed. With "Cancel" the print job is canceled.

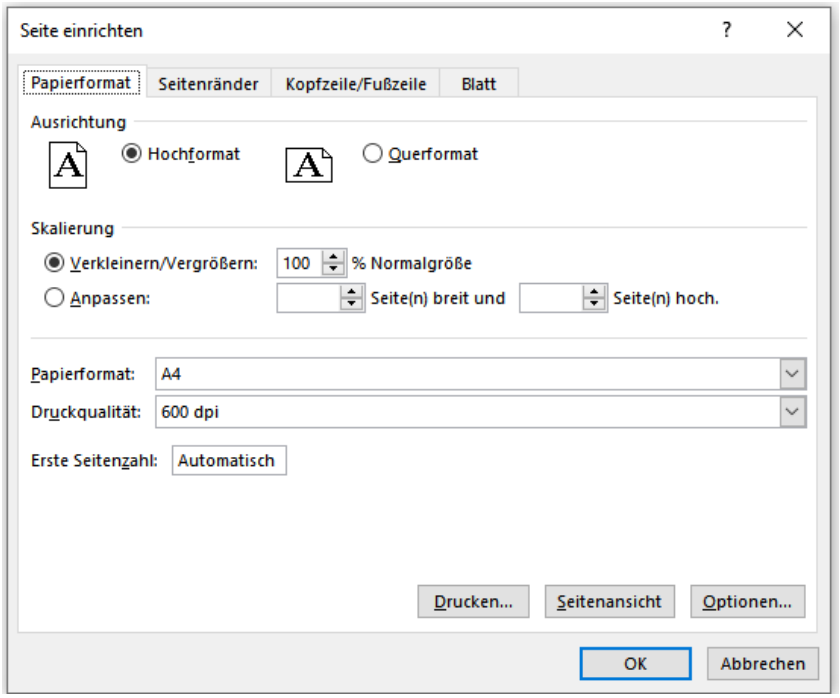


Note:	The print preview is not available in Inplace.

2. ShowPageSetupDialog

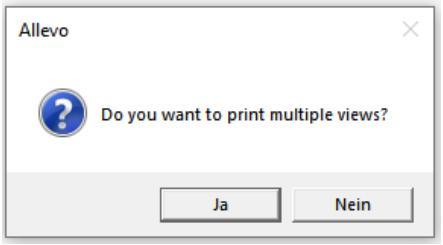
With the dialog "Page setup" all page details can be predefined before printing. Exception: the repeat rows above are determined by VBA (if a freeze is set up).

Since you can only make settings for the respective worksheet with this dialog, the ShowPageSetupDialog option is **only available for printing the active view (PrintView), but not for PrintMultipleViews.**



3. ShowPrintViewsWarning

An additional query can be made before printing multiple views with PrintMultipleViews.



20.2 CadoPrintViewSetup

Additional customer-specific settings can be programmed via the Cado CadoPrintViewSetup.

When is the Cado useful?

- If print settings are to be forced. Because despite sheet and folder protection, it is not possible to prevent settings from being changed via the Excel page setup dialog. (Offering the ShowPageSetupDialog option would be counterproductive in this case).
- If settings via the Excel page setup dialog are not possible (e.g. save date or cost center in the footer or header).

21 StopPlanning

With StopPlanning, you can cancel planning according to SAP depending on a flag and output a language-dependent message. (Previously there were customer-specific solutions for this with CadoSapButton-Pressed).

If a CadoSapButtonPressed is implemented, then StopPlanning is not executed, i.e. the customized solution always wins, so that the migration of older Cados should not be a problem.

How it works

A one-cell name range is set up (name freely selectable), which indicates whether saving to SAP is permitted or not. The following applies: 0 = it is saved, 1 = planning is canceled and a message is displayed

How to set it up

This is set up in the CustomizingBase table:

Customizing

Process	SubProcess	Parameter	Key	Description	Value
StopPlanning	FlagRange				MyRange
StopPlanning	Message	D			Abbruch
StopPlanning	Message	E			Break

1. Process: StopPlanning
2. SubProcess
   - FlagRange
     - Parameter: (not used)
     - Value: Name range of the cell that is to be checked for 0 or 1.
   - Message
     - Parameter: Language
       If no parameter is specified, the message is used for all languages. (The relevant language here is not the Excel language, but the ABAP language (inplace) or the language that is entered in the GlobalInfos (offline)).
     - Value: Text for Message

Best Practice

If you do not know which languages can occur, you should always specify a variant without a language parameter. This then applies "for all other" languages.

Customizing

Process	SubProcess	Parameter	Key	Description	Value
StopPlanning	FlagRange				MyRange
StopPlanning	Message	D			Abbruch
StopPlanning	Message	F			Abort
StopPlanning	Message				Break

22 Dictionary

22.1 Activation of the Dictionary (TranslateSheets)

With the Dictionary function, a possibility has been created to translate any text entries depending on the logon language. For this purpose, the translations are stored at a central location in the master, namely on the |DICTIONARY| worksheet.

To activate the dictionary, the function "TranslateSheets" must be entered under a suitable event in the [CUSTOMIZING] sheet under [User Exit]:

User Exit

Basic		Macro	
Event	View	Macro1	Macro
OPEN_IN_SAP	Main01	TranslateSheets	

Activation of the Dictionary by the event "Open in SAP" (OPEN_IN_SAP)

Usually event OPEN_IN_SAP is used, alternatively also AFT_READ, if the contents in the dictionary arise dynamically (read from satellite).

22.2 Functionality of the Dictionary

All text entries of the master and the associated translations are recorded and managed centrally for the desired languages in a list on the |DICTIONARY| sheet. The list can be maintained manually in the master or filled when Allevo is started (e.g. via satellite 0, table [DictionaryBase]):

DictionaryBase

Index	E	D
1	Activity Dependend Planning	leistungsarten-abhängige Planung
2	Activity independend	leistungsunabhängig
3	Activity Input (Partner Object)	Leistungsaufnahme (Ursprungsobjekt)
4	Activity Input (Source Object)	Leistungsaufnahme (Partnerobjekt)
5	Activity Output	Leistungsabgabe
6	Activity Type	Leistungsart
85	Jun	Jun
86	Jul	Jul
87	Aug	Aug
88	Sep	Sep
89	Oct	Okt
90	Nov	Nov
91	Dec	Dez
92		
2	E	

1

2

Dictionary entries

The column key describes the language according to the nomenclature on SAP page, i.e. D for German, E for English, F for French etc.; see row (1) in the figure above.

The starting language is to be entered in the result row of the table below (2). To get this information also when filling the data via satellite, there is a separate, single-cell area {DictionaryStartLanguage} for this from Master Version 3.4.6.

When the "TranslateSheets" function is triggered, the texts of those cells in the workbook are translated that have "KernHeader1...n" or "KernHeadline1...n" as their format template. It is translated into the language that was passed in the global parameter Language from SAP as the logon language to the Excel file.

Translation takes place exactly when the entry in the correspondingly formatted cell in the Dictionary has an entry in the "Language ID" column. Thus, if a cell in the master contains the entry LART, this entry can be translated into either "Activity" or "Service" depending on the language.

The following rules apply:

Entries determined by formula are not translated.

If no translation is found for an entry in the Dictionary column "Language ID", then the entry is not translated or overwritten.

If the language from the global parameter Language is not found in the dictionary, translation is done into the default language (in the default master this is English). If this is not found either, no translation takes place.

Note:	For headings and titles that should NOT be translated, the styles "KernTop1" to "KernTop3" can be used.

Performance (especially from Excel 2013)

By default, Allevo takes all sheets into account when searching for translation-relevant texts (search on all sheets for relevant content/styles). The list of relevant sheets can also be explicitly specified via table [CustomizingTranslateSheets] (already created on the "Dictionary" sheet in the standard master).

Especially under Excel 2013 and successor versions this can improve performance, because in these Excel versions unprotecting and setting sheet protection is slower than before.

23 The summary sheet and the total sheet (TotalSheet)

23.1 Function overview

When working in the MultiPage mode of Allevo, a comprehensive display is often desired in order to see overall totals per cost element, etc. across all objects contained in the file (i.e. also totals across cost centers, orders, WBS elements).

Allevo has two basic forms of display types for this:

A summary sheet with the same cost element structure as the individual sheets of the MultiPage file, where the Excel formula SUMME adds the values per cost element over all sheets.

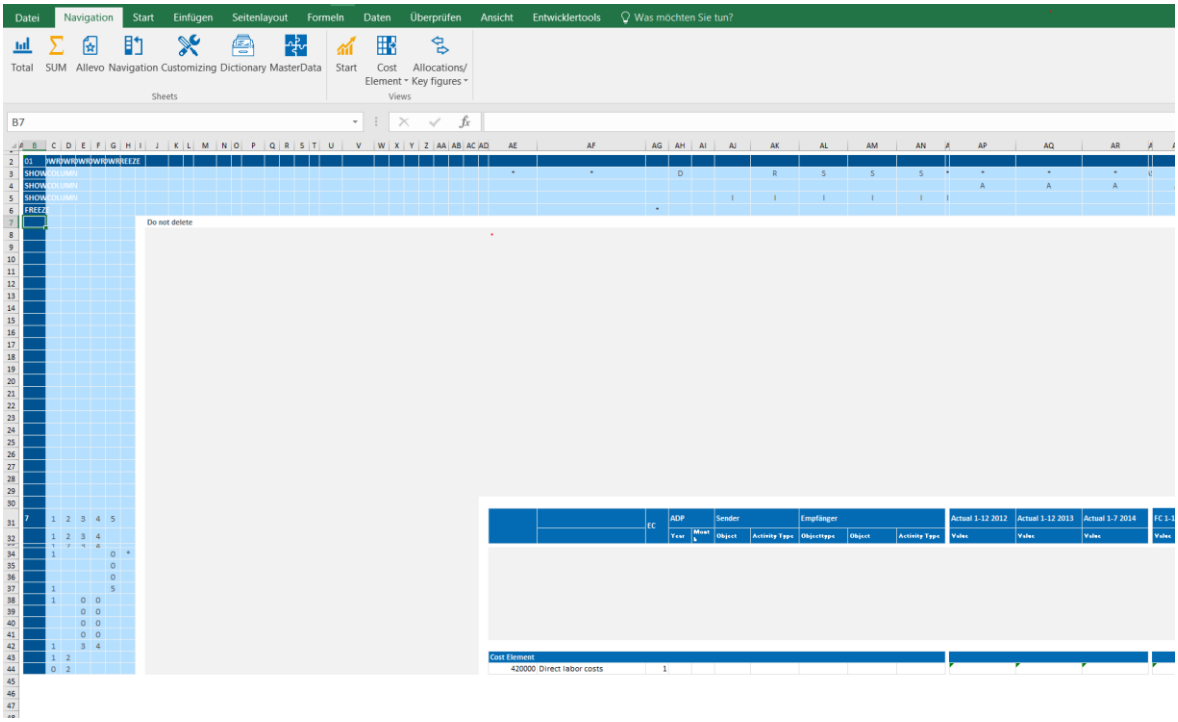
A macro-based total sheet (overview-summary sheet) that bears similarity to the classic operational accounting sheet.

23.2 Sum sheet (SUM)

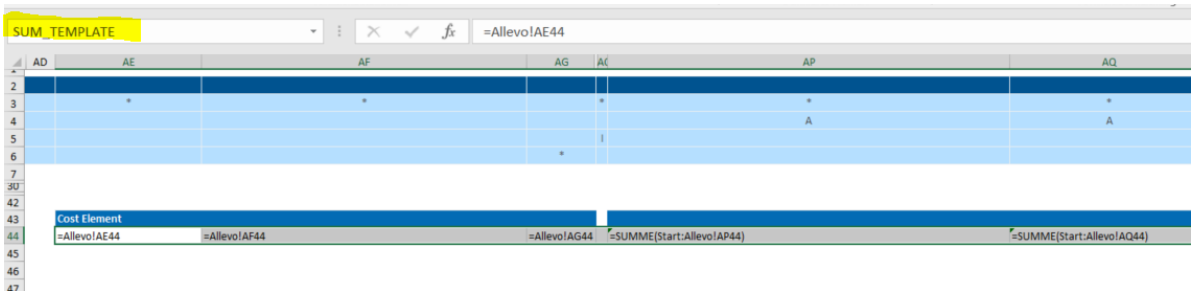
Area	Feature	Info
Impact Area		
Area names	SUM_TEMPLATE	The SUM_TEMPLATE area is used to determine the sum sheet
Controls		
Macro	CreateSumSheet	
Settings		
Area names	CY_KEYTOTAL	Rows in the [Standard] that are transferred to the totals sheet
Configuration table	Customizing	
Code		
VBA	basKitSum	

The Totals sheet uses the Excel formula SUMME, which can not only add rows or columns, but also pulls the sum of a cell over a defined sequence of sheets. When setting up the sum sheet, the following steps are necessary:

- Create an empty spreadsheet in front(!) of the default sheet named |Start|. This sheet can be hidden afterwards
- Inserting a copy of the |Standard| sheet named |SUM|
- All corners of the |SUM| sheet are removed, except for the navigation, which can then be adjusted if necessary. Only the areas are deleted, not the rows and columns, so that the data coordinates still correspond to those of the |Standard| sheet.



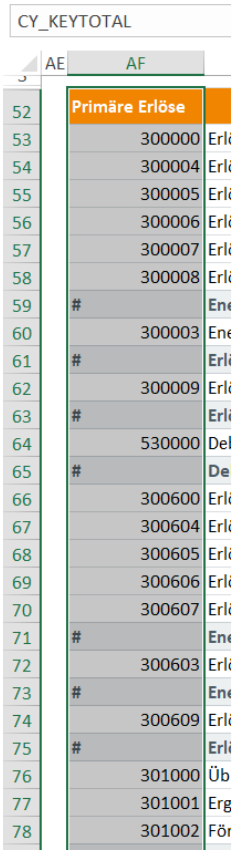
- The range name {SUM_TENPLATE} is placed over the first data row in the |SUM|.



- All further data rows can be deleted

- Now a formula is entered in all value fields of the data row according to the following scheme: **=SUM('Start:Standard!A1)**

- In |Standard| the area name {CY_KEYTOTAL} is placed over the cost elements with totals hierarchy structure. The area must include the first header row:



- The *CreateSumSheet* macro then creates the further row structure and copies the sum formulas accordingly.

- Formats for formatting and recognizing totals rows are entered in the [Customizing] table:

Customizing

Process	SubProcess	Parameter	Key	Descri	Value
SumSheet	SumStyle	RowType	1		#
SumSheet	SumStyle	RowStyle	1		KernSum01
SumSheet	SumStyle	RowType	2		##
SumSheet	SumStyle	RowStyle	2		KernSum02
SumSheet	SumStyle	RowType	3		###
SumSheet	SumStyle	RowStyle	3		KernSum03
SumSheet	SumStyle	RowType	4		####
SumSheet	SumStyle	RowStyle	4		KernSum04
SumSheet	SumStyle	RowType	5		#####
SumSheet	SumStyle	RowStyle	5		KernSum05
SumSheet	SumStyle	RowType	6		######
SumSheet	SumStyle	RowStyle	6		KernSum06
SumSheet	SumStyle	RowType	7		#######
SumSheet	SumStyle	RowStyle	7		KernSum07

- Format the summary sheet according to your own ideas. As a rule, the comment column and the satellite areas on the summary sheet can be deleted, as experience has shown that they are not used.

Note:	After the totals have been inserted in the totals sheet, columns and rows in the totals sheet can also be deleted. However, if the column or row structure in the |Standard| sheet is changed, the totals sheet must be adjusted from the respective point.

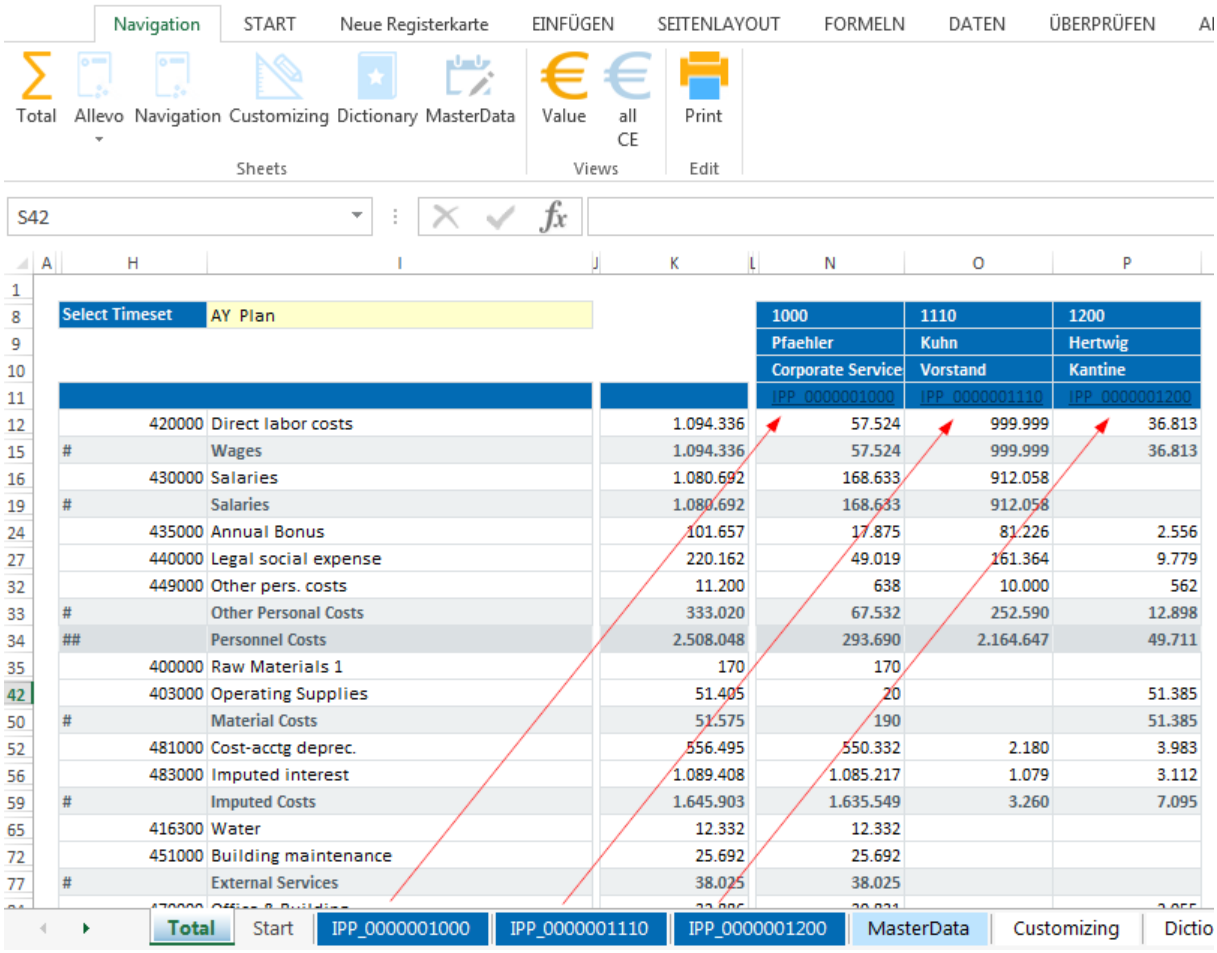
When copying the |Standard| sheet, its name ranges are also copied. Although this basically does not affect the runnability of Allevo, the name ranges on the summary sheet should be deleted again.

When copying the |Standard| sheet, the navigation function is also copied. If necessary, it should be adapted for the summary sheet.

23.3 Macro-supported total sheet (overview total sheet)

Basically, the macro-based total sheet corresponds to a classic overhead allocation sheet: the cost centers (orders/WBS elements) selected in the Allevo Multi are displayed in columns and the cost elements are displayed in rows.

Thus, an aggregation of read and plan columns of different spreadsheets takes place here, whereby a selection list can be used to control which data (year, version, etc.) of the cost centers are to be displayed next to each other.

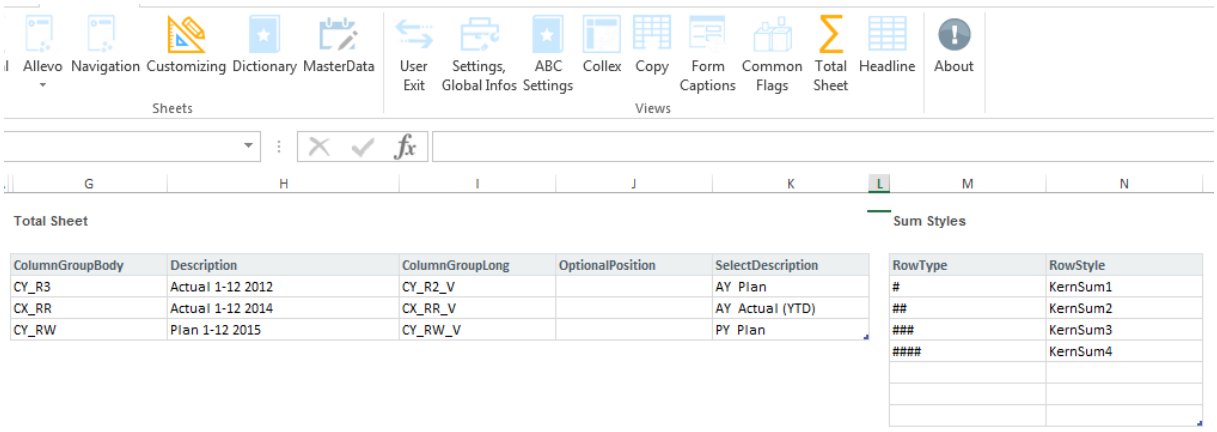


Main view total sheet with data

The total sheet is designed to be set up with little effort. Usually it is sufficient to adapt the number of rows and the formatting to the |Standard| sheet.

Note: The macro-based total sheet is part of the standard Allevo master and cannot be created manually like the simple formula-based total sheet.

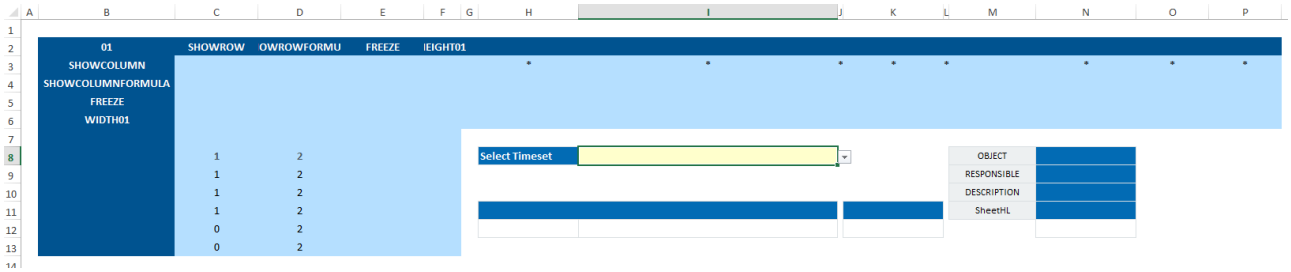
The assignment of the displayed columns in the total sheet is done by naming the read columns or their absolute column number (OptionalPosition) in Excel. In addition, a description for the respective entry in the dropdown field should be assigned under Select Description.



23.1 Configuration total sheet

The mappings made in the configuration allow the Total sheet in the template to be left free of any row and column structure, so that the user can concentrate on configuring the navigation and does not have to maintain the row structure twice. The row structure is taken from the {CY_KEYTOTAL} namespace of the |Standard| sheet).

The gray entries OBJECT|RESPONSIBLE|DESCRIPTION can be replaced by any local parameters (see info sheet) to display further information from the objects.



Blank total sheet in template

24 Additional functions in Allevo Master

24.1 Save any object-relevant data in SAP

During planning, it can be helpful to manage any additional data for the object and have it available again the next time planning is called up (e.g. for secondary calculations). For this application, an area can be set up in the Allevo master whose data is automatically saved in an SAP table and later read from there again. The data exchange takes place optionally with reference to layout or only with reference to object (but generally without reference to year and / or version).

Note:	Of course, a satellite could also be set up for this kind of requirement (possibly also without reference to year and version). Disadvantage: the structure of a satellite table is largely fixed and must first be set up in the SAP system via an append. The solution here works completely without any preparatory work in the SAP system.

A structured table with the name [ObjectFields] must exist on the Excel side (in earlier Allevo versions this table was called ZZObjectFields): Allevo takes all data in this table into account when writing and reading. In addition, it should be noted:

The structured table must be created with a heading and this heading must not contain spaces (to avoid error messages from Excel).

The table must initially have at least two rows.

Since the contents are object-specific, the table must be created on the template sheet (i.e. usually the "Allevo" sheet).

Any data can be entered in the structured table area: there is no default related to the data type. The number of data records can also be changed as desired at runtime; Allevo automatically adjusts the length of the table.

Note:	On SAP side further setup steps are required: the data is automatically saved to the initial object and to the current layout. Optionally constant NO_LAYOUT_FOR_FIELDS is to be applied if the data is to be saved without reference to the layout.

The table contents are transferred to SAP in XML format and stored in database table /KERN/IPPFIELDS. The contents cannot be evaluated on SAP side.

24.2 MODULE: ReportingKit

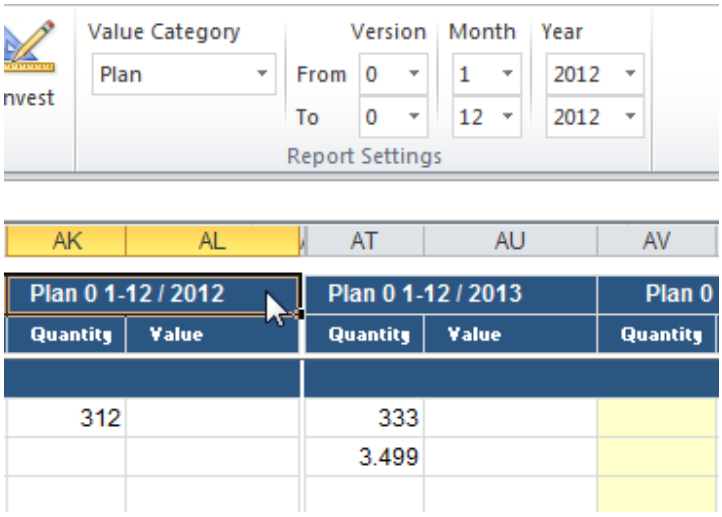
Using the function described here, it is possible to change central characteristics that describe the content of the displayed columns (i.e. year or version, for example).

The module is activated via the *AcitvateReportKit* setting in the Customizing sheet.

Note:	This function was originally intended only for Allevo's reporting mode (i.e. when starting the /ALLEVO/KSREP transaction for cost centers, for example). However, it can also be used in

	planning, e.g. to temporarily access actual data from another year. However, changing plan columns can be critical depending on the selection combinations.

If suitable parameters are stored in the Customizing of the master (see below), the user will see input fields with the characteristics of a column definition in the "Navigation" ribbon.



24.1 Change column definition information individually

The contents of the fields offered always refer to the column that was previously marked with the mouse (click on header cell). If an entry is changed (e.g. for a change to another year), Allevo deletes the data in this column; in addition, this column is highlighted in color until reference data is read in again.

Note:	To achieve this marking of the column, the relevant header cells must have "KernHeadlineParameter" as a style sheet (simply create a copy of KernHeadline1 and assign the desired color there).

Which values the user can select must be defined in Customizing via a table [ZZCustomizingHeadlineSelection] (with header line).

ValueCategory	CategoryDescription	VersionFrom	VersionTo	PeriodFrom	PeriodTo	YearFrom	YearTo
1	Ist	0	0	1	6	2005	2005
2	Plan	1	1		12	2006	2006
3	Obligo	2	2			2007	2007
4	Budget					2008	2008
						2009	2009

Parameters for column definition: Restriction of selectable values

The parameters of this table define which contents the user can change via the ribbon: in the example above, for example, the entries for the year can be changed to values between 2005 and 2009. The table is interpreted column by column: in the example, the start period is always fixed at 1; for the to period, either 6 or 12 can be entered.

Only if this table is present, the input fields shown above are available in the "Navigation" ribbon. Only the columns that are present in the table and have a value there are available for selection (so also pay attention to the correct spelling of the column headings).

Note:	Additionally, the constant DYNAMIC_COLDEFS must be active on the SAP side, it determines whether changes are taken over from the Excel side (e.g. depending on the reporting mode, see documentation on the constant). In addition, please note the following restriction: on the SAP side, no relative reference may be active in the column definitions.

24.3 Customer-specific VBA extensions in Allevo Master

Allevo projects are Excel projects: the flexibility of Excel makes it possible to build a master in such a way that the customer's desired design requirements are met (e.g. using Excel formulas or via Allevo navigation).

In individual cases, however, it may be useful and necessary to store customer-specific functions as VBA coding in the Allevo master (e.g. for mapping special input options during planning). Such solutions can often be called up like individual macros and therefore do not affect the other functions that are stored in the Allevo master.

In other cases, however, it may be necessary to extend basic functions of the Allevo master itself. Two examples of such a requirement:

Individual functions in data communication between SAP and Excel (see use case below for checking completeness of data).

Customize navigation with ribbon enhancements.

Custom PDF output.

In order to enable such functions, the VBA code of the Allevo master contains jump points where customer-specific code can be added (short name "Cado", similar to the UserExits in the SAP system).

These enhancements are usually made in the course of an Allevo implementation project.

SAP Excel communication application example:

Cado jump points are stored for all SAP commands that can be called from the inplace processing; e.g. for command PLANNING when calling button "Accept plan data" (also applies to a customer button that has been activated via constant BUTTON_CUST1). This way, e.g. the completeness of plan data on Excel page can be checked before this data is transferred to SAP. Important for this use case is registration at the time OPEN_IN_SAP via *CadoSapButtonPressed*.

24.4 Optional VBA extensions in Allevo Master

Since not all functionalities are required in every customer project, some modules are not in the master by default. This means that the master is not overloaded with code that may not even be needed. If necessary, the respective module can simply be inserted and integrated in the master.

Some of these functions are already mentioned in the sections above. Such functions are usually aligned individually to customer requirements in the implementation project.