EasyInput

a more effective user work due to integration of SAP and MS Excel

Business Application Programming Interface (BAPI) and Function Scripts

SAP Gold Partner

BCC Data Centers

BCC EXTRA

- EasyInput is one of the products from the **BCC EXTRA offering**.

- BCC EXTRA is a set of unique products allowing you to boost the effectiveness of the use of SAP systems at low cost. These are functionalities developed in-house by BCC, enhancing the SAP system standard, and the tools facilitating the SAP system implementation and development.

- We build EXTRA products based on a many-year implementation experience gained while working for different customers from Poland and abroad.

More BCC EXTRA products at: **www.bccgroup.eu/extra**

Business Application Programming Interface

- **BAPI** (Business Application Programming Interface) is a precisely defined interface providing access to processes and data in SAP systems. BAPIs are defined as API methods of SAP business object types. These business object types and their BAPIs are described and stored in the Business Object Repository (transaction BAPI).

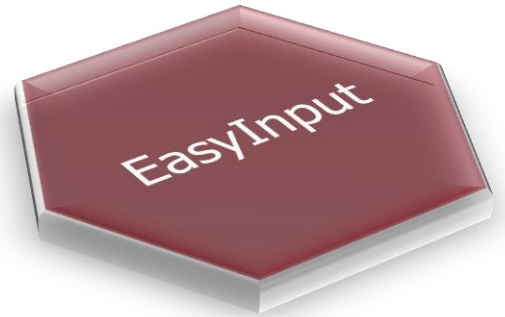
- SAP provided the first BAPIs for customers and external providers in SAP ERP release 3.1, enabling them to integrate their software components with the SAP System. The number of BAPIs is increasing with each SAP system release and with this the extent of object-oriented access to the SAP System.

EasyInput – Use of BAPI

- SAP business objects are accessed through BAPIs (Business Application Programming Interfaces), **which are stable, standardized methods**. SAP business objects and their BAPIs provide an object-oriented view of SAP system business functions.

- BAPI methods are implemented as remotely accessible **function modules** in the SAP Netweaver System. **EasyInput can** call such function modules and **execute BAPI methods**.

- Function scripts are less dependent on the given system settings than transaction scripts. They do not require adjustment when system is upgraded (which can influence transaction scripts). Yet, they can be more difficult to create for business users as they sometimes require programming skills.



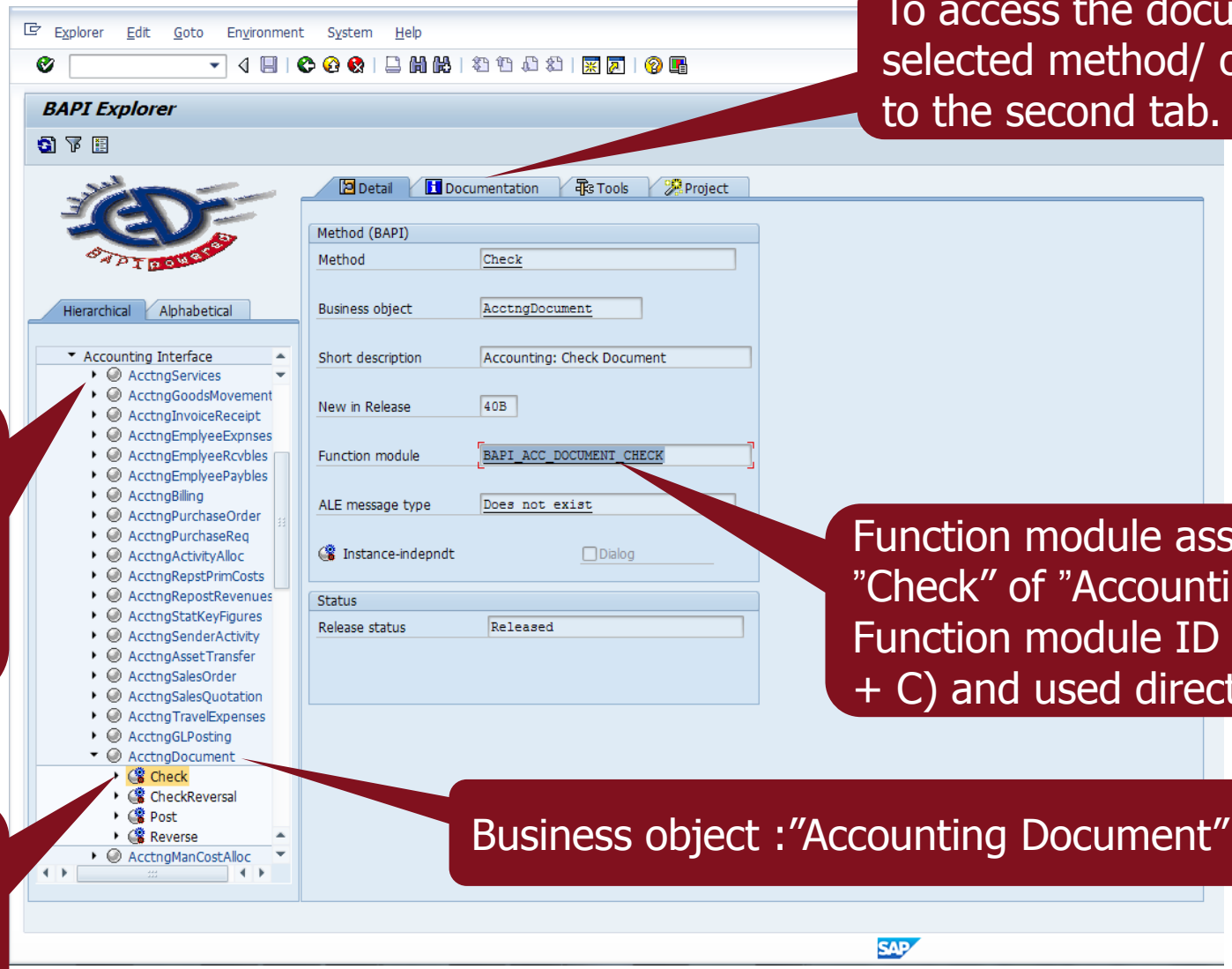
Business Objects Repository – transaction BAPI

- The **Business Objects Repository (BAPI transaction)** is used to access information about all BAPI objects and methods accessible within an SAP system.

- The best approach is to log into an SAP system in English in order to browse the repository efficiently, as most of the interface documentation is prepared (and is only accessible) in **English language**.

- In Business Objects Repository one can find predefined by SAP business methods (e.g. for master data / document creation; for reading data from SAP objects) and their documentation. The function modules assigned to BAPI methods can be found as well in Business Objects Repository.

Business Objects Repository – finding relevant function modules



To access the documentation of the selected method/ object one can switch to the second tab.

Business objects and their methods are accessible via component hierarchy or alphabetically.

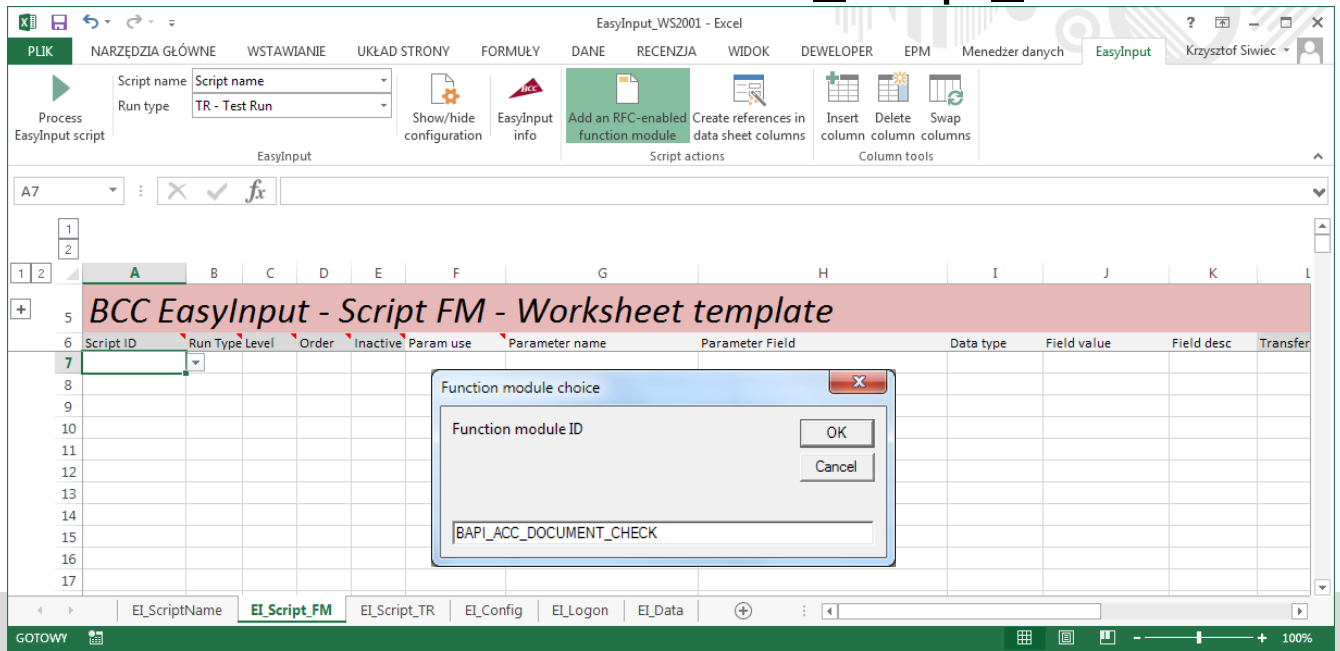
Function module assigned to method ”Check” of ”Accounting Document”. This Function module ID can be copied (Ctrl + C) and used directly in EasyInput.

Method ”Check” (checking posting) of business object ”Accounting Document”

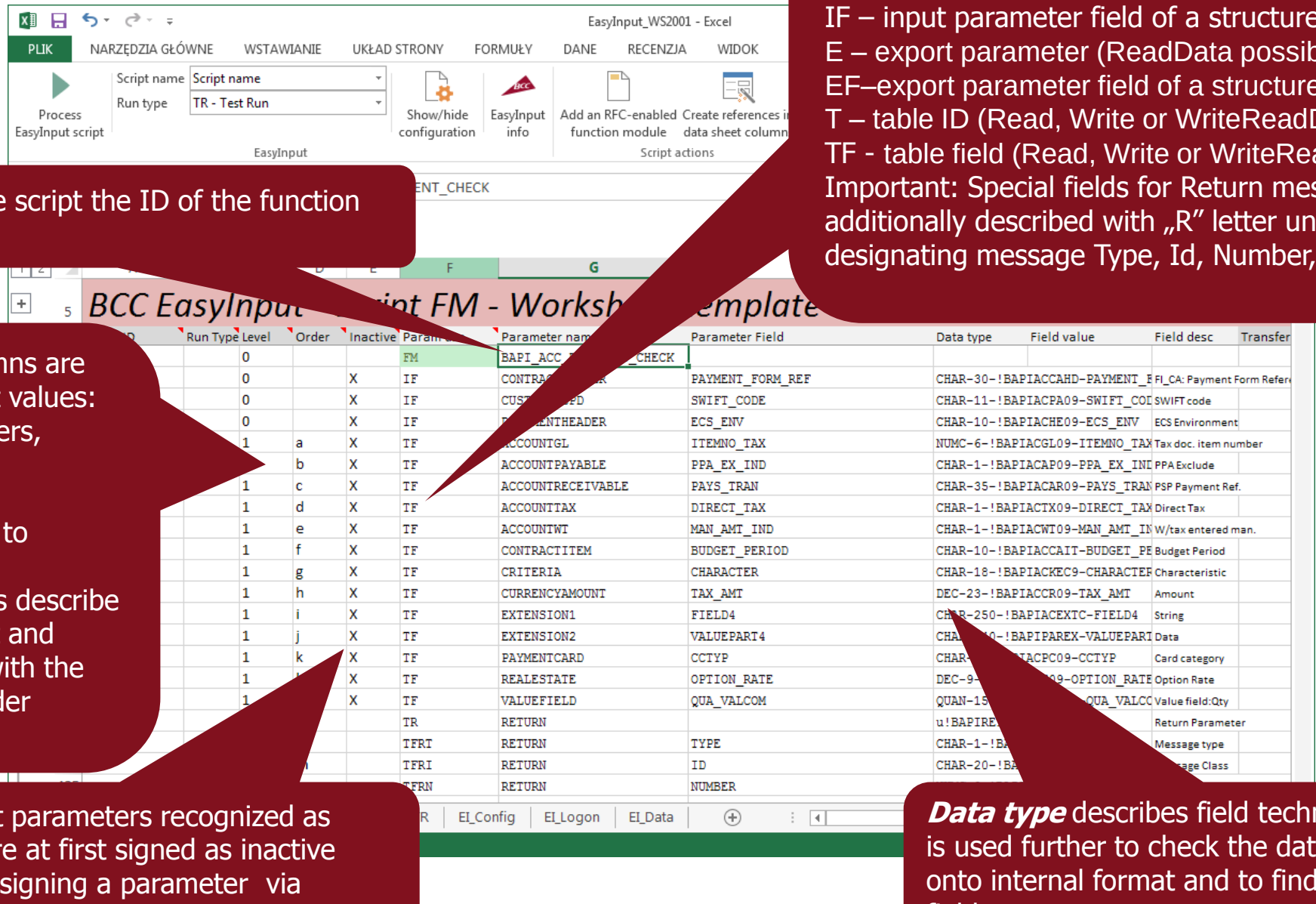
Business object :”Accounting Document”

Importing Function Module Interface to EasyInput

- After selecting the right function module in Business Objects Repository (transaction BAPI) one can import the function module into the EasyInput workbook.
- Remember to enter the SAP system credentials on the EI_Logon worksheet beforehand, as these will be needed to import function module interface information from the SAP system.
- The function module interface can be imported in EasyInput workbook by pressing the *Add an RFC-enabled function module* icon on the EI_Script_FM worksheet.



Importing Function Module interface



At the beginning of the script the ID of the function module is put.

Param use describes how the script row is used:
I – input parameter (WriteData possible)
IF – input parameter field of a structure (WriteData possible)
E – export parameter (ReadData possible)
EF–export parameter field of a structure (ReadData possible)
T – table ID (Read, Write or WriteReadData possible)
TF - table field (Read, Write or WriteReadData possible)
Important: Special fields for Return messages are additionally described with „R" letter und additional letter designating message Type, Id, Number, Message text.

The level/order columns are assigned with default values:
0 for simple parameters,
1+letter for tables

This can be adjusted to business needs (e.g. sometimes two tables describe the same data object and should be assigned with the same letter in the order column)

All parameters (except parameters recognized as RETURN messages) are at first signed as inactive („not used"). When assigning a parameter via WriteData or ReadData the X should be removed !!!

Data type describes field technical information, that is used further to check the data, to convert the data onto internal format and to find the description of the field.

Actual and Test Mode for Function Scripts

- In some cases not committing the test run (the default EI behavior) will be enough to differentiate between test and actual run. In this case all the script lines can be shared (Run Type =”C”). This could be used where the system does not assign internal numbering to documents or master data. When internal numbering is created such a solution could lead to not consecutive numbering (holes in number ranges).

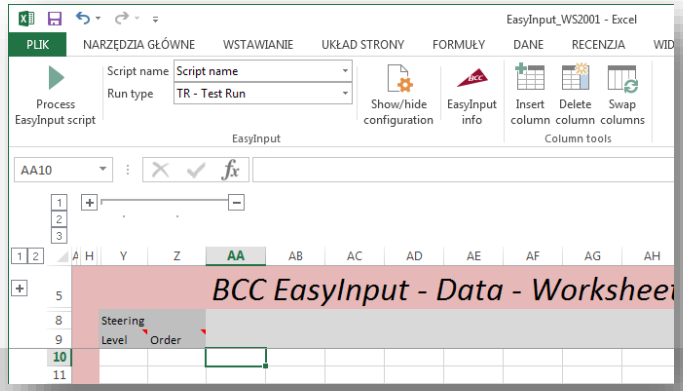
- Some financial BAPIs use modules with ending _CHECK for testing and _POST for actual run. In these cases different function modules can be used. Since they have only small differences in parameters most of the script for both modules can be shared (”C”). The exception of this rule will be lines with function module ID (one with ”T” and one with ”A”) and additional lines with returned document number for actual run (”A”).

- Other BAPI function modules have one of the parameters to be filled with „X” designating the test run. In such case all the script is shared (”C”) and this single line with the parameter is used only in test run (”T”)

- There are also BAPIs for which there is no test mode and are committed automatically. In such cases resignation of the test mode (TestModePresent) on the EI_Config worksheet is possible.

Level/ Order Columns in EasyInput Script

- If a document has further sublevel for each line item (e.g. conditions in SD for each line item) next level number can be used (2 in this example)
- **Order can be used if two different sublevel exists** (e.g. Roles and Profiles for a user in transaction SU01). Than one sublevel can be assigned level/order pair 1/a and the second 1/b.
- The level/order pair defined on the script is usually automatically assigned to data rows on EI_Data worksheet (Y/Z columns). The system use linking data column with script (defined in script) to detect the right level/order before processing the data. For some scripts however, where the data columns from various level/order overlap EasyInput may not be able to assign data row to a given level/order. Then manual assignments in columns Y/Z of data row is needed.



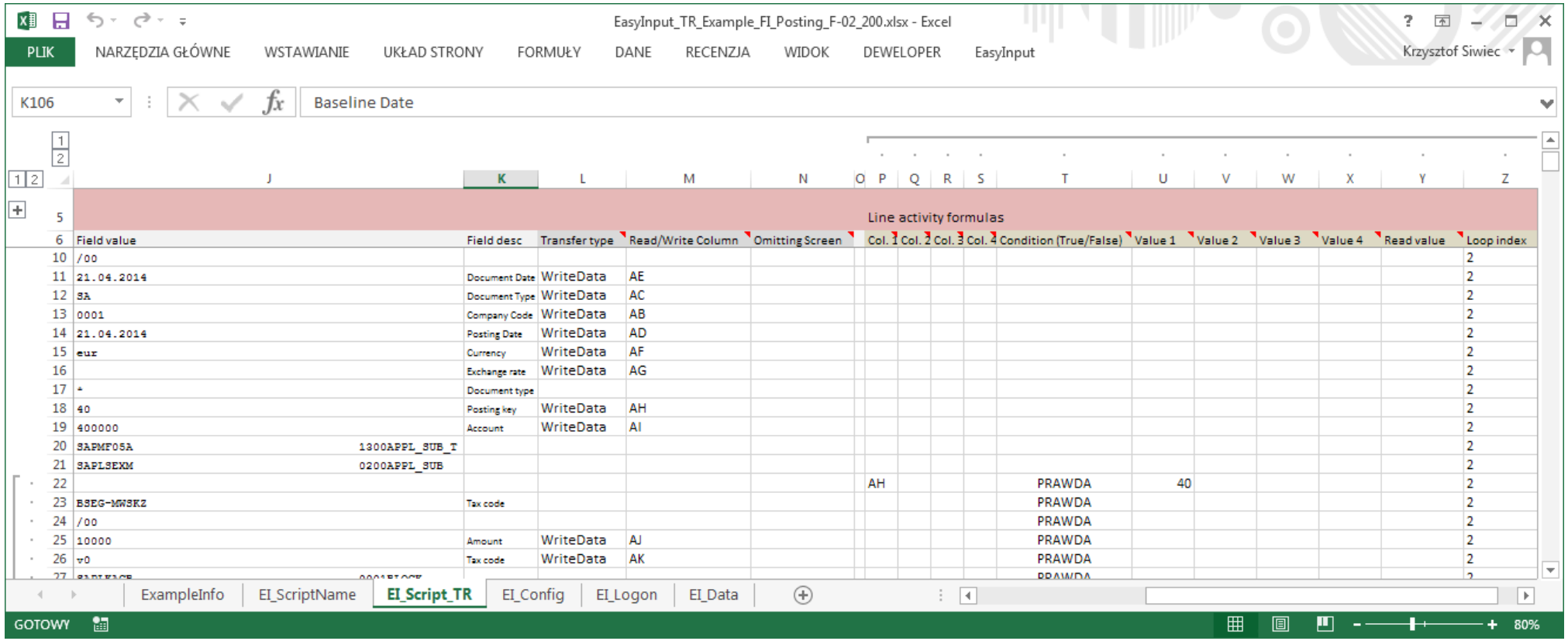
Function Scripts with More Than One Sublevel

- Many BAPIs require more than one sublevel of data passed as import parameters. For accounting BAPIs such a need arises when passing CO-PA assignment for line item.
- Example of lines sequence on data worksheet:
  - Level 0 header
  - Level 1 line item (variuos types GL Item/ Customer/ Vendor/ Tax, etc)
  - Level 2a CO-PA characteristics
  - Level 2b CO-PA value fields

- When creating such a script it is necessary to follow one golden rule:
- The sequence of level/order pairs in the script worksheet should follow the sequence in which data will be given on the data sheet. If one does not obey the rule some data lines may be omitted! Please reorder the scipt lines accordingly. The sequence of a pairs on the same level is irrelevant (e.g.2a/2b in the script can be switched in order), but the sequence of sublevels vs the level above them (2a/2b should follow in the script level 1) is important!. Also script order/level pairs should be conignuous (the only exception is level 0, that does not have to be continuous). Please switch debugging on when developing the script to see whether all the data lines are taken into account.

Line Activity Formulas 1/4

- **Line activity formulas** is the name of an additional functionality contained in EasyInput, that is configured in columns P-Z and sometimes used in column J (field value) of the EI_Script_XX worksheet.



Line Activity Formulas 2/4

- Line activity formulas are used to:
  - **Make a certain script line conditional (T)**
  - **Calculate a value of the field value column (J)**
- Line activity formulas can be based on:
  - Data contained in the data worksheet (by default EI_Data)
  - Technical data contained in column Z - Loop Index
  - Data read in the script (ReadData option, Y)

- Referencing data contained in the data worksheet

  Each script line is assigned to a certain level/order. When processing a Data line assigned to the a level/order all script lines with the same level/order are processed from top to bottom in the active script. For each script line if columns P/Q/R/S are filled with references to columns in Data sheet (e.g. AA, AB,…), relevant data sheet values are copied to columns U/V/W/X. Thus one can create an Excel formula in the line based on data in columns U/V/W/X, as these columns are filled with data in runtime. Remember that, the script is processed from the top down, so if one reads a value in the first line of a certain level/order one can reuse it in formulas in the lines below with the same level/order or with higher level (without filling the references in columns P/Q/R/S in these lines).

  Important: From version 2.17 B1009 on it is possible to read data row number. Just put string ROW in columns P/Q/R/S.

Line Activity Formulas 3/4

- Referencing the technical data contained in column Z - Loop Index

  Sometimes in the formula one has to make a distinction between the first document line and the others. For this purpose Loop Index value is filled in column Z of the script worksheet in the runtime. Loop Index column contains for lines on level 0 the number of consecutive transaction processed, and for the lines on higher levels, the number of a consecutive data line (e.g. Line item number) processed within a transaction.

- Referencing data read in the script (ReadData option, Y)

  If a script contains several transactions/ function modules then the result read in one can be used in the next. Apart from copying the data read with the ReadData statement to the Data worksheet, the data is also copied to column Y of the script worksheet to the relevant line. Important! Since reading data is performed **after** the transaction/ function module is called, its result from column Y cannot be used when a script contain only one transaction/function module!

- Make a certain script line conditional (T)

  Apart form static inactivation in column E of the script, a dynamic inactivation of the line can be performed in the column T. If the script is not inactivated in column E, then:

  - If there is no formula in column T, the script line is active
  - If there is formula returning true in column T, the script line is active
  - If there is formula returning false in column T, the script line is inactive
  - If the script is inactivated in column E, than the line is inactive and no other checks are performed.

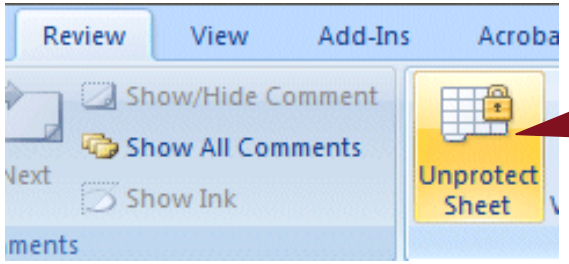
Line Activity Formulas 4/4

- Calculate a value of the field value column (J)

  Value passed to script line can be passed as:

  - A constant put into J column
  - A variable read from the Data worksheet with the statement WriteData and the reference to the data column (L/M)
  - A formula put into J column

  In the third case one has to take off the script worksheet protection and change the formatting of the cell to general, so that Excel treats the formula as the formula and not as a string. In the formula one can reference the fields filled in the runtime in columns U/V/W/X and in the special cases in column Y.

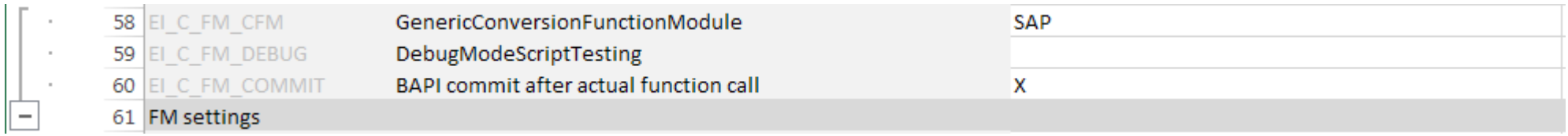


Use standard MS Excel ribbon menu to unprotect a worksheet.

BAPI Commit

- Newer BAPI functions module do not update data tables directly. Instead they gather changes and execute them when BAPI function BAPI_TRANSACTION_COMMIT is called.

- In order to reflect this functionality on EI_Config worksheet in the section FM settings there is the switch „BAPI commit after actual function call”. When it is on („X”), then after each BAPI function module call for actual run, the BAPI_TRANSACTION_COMMIT is called as well. For older BAPIs it can be switched off for performance reasons.



58	EI_C_FM_CFM	GenericConversionFunctionModule	SAP
59	EI_C_FM_DEBUG	DebugModeScriptTesting	
60	EI_C_FM_COMMIT	BAPI commit after actual function call	X
61	FM settings		

Function Script Testing

- Once the function script is created, it is possible to test it. By testing, one can check which parameters and with what values are passed to the function module). In order to switch on the testing mode set the „DebugModeScriptTesting” configuration switch on the EI_Config worksheet.

- In test mode during script run the additional worksheet EI_Debug is created and all the parameters passed to the function module are shown there. The parameters shown there, are presented BEFORE conversion to the internal format needed for the function module calls.

58	EI_C_FM_CFM	GenericConversionFunctionModule	SAP
59	EI_C_FM_DEBUG	DebugModeScriptTesting	
60	EI_C_FM_COMMIT	BAPI commit after actual function call	X
61	FM settings		

Function Script Parameter Conversion

- Each parameter value passed to the function module should be converted from the external format (shown to the user) to the internal format of data. This is normally done via conversion routines attached to ABAP data elements.

- Normally to be able to convert any field the user needs the authorization:
  OBJECT: 'S_TABU_RFC'
  ACTVT: '03' (=display)

- When using full version of the EasyInput tool one can use BCC function module instead of the standard SAP one (switch on the EI_Config worksheet). Using the BCC conversion function module does not require the authorization object stated above.

58	EI_C_FM_CFM	GenericConversionFunctionModule	SAP
59	EI_C_FM_DEBUG	DebugModeScriptTesting	
60	EI_C_FM_COMMIT	BAPI commit after actual function call	X
61	FM settings		

www.bccgroup.eu/extra

Free trial version
Order online
Other BCC EXTRA products