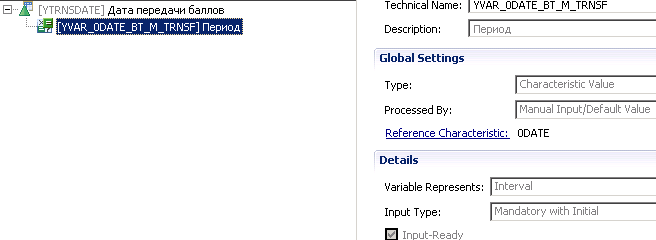
[Содержание](#Содержание)

[Customer Exit Variables](#Customer_Exit_Variables)

[BEx with input parameters](#Bex_with_input_parameters)

# [Passing Input Parameter from BEx to CV](#Passing_Input_Parameter_from_BEx_to_CV)

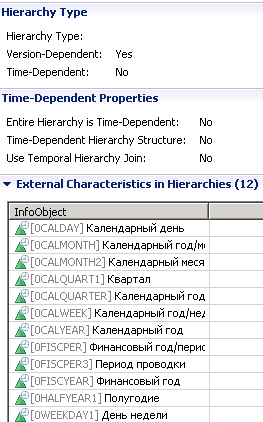
**Интервальная переменная**



Где *YTRNSDATE*



*Иерархия*



**Создание переменных в SAP BW**

− Помогите разобраться с созданием текстовых переменных, которые я хочу видеть в отчёте.   
Для объекта 0CALMONTH созданы две переменные variables *Processing by Customer Exit*

* per\_1 - заполняется с экрана при запуске отчёта / создана как *is ready for input* /;
* per\_2 - вычисляется по алгоритму описанному в EXIT\_SAPLRRS0\_001 в зависимости от переменной per\_1.

Всё вычисляется как надо. Для обеих этих переменных надо создать текстовые переменные, которые будут присутствовать в названии полей структуры. Для per\_1 сделала текстовую переменную и она отображается в отчёте. А вычисляемой per\_2 - проблема. Если её создавать как *Processing by Replacement path*, то в списке переменных Variable нет нужной мне per\_2. Что не так?

− Создание текстовой переменной для per\_2 нужно сделать в EXIT\_SAPLRRS0\_001 / аналогично тому, как вы создавали переменную per\_2/

per\_1 - готова к вводу / *is ready for input/*

per\_2 = F1(per\_1) - USER-EXIT

per\_text\_1 = замена per\_1

per\_text\_2 = F2(F1(per\_1)) - USER-EXIT

− В системе нигде не описана математика стандартных переменных, а иногда хочется посмотреть на стандарт, скопировать и поправить под себя. Например, измерение 0CALMONTH содержит интересную переменную 0CML3CM *Последние 3 месяца включая текущий*. А мне нужно сделать такое же, только не включая текущий.

Подсмотреть алгоритм стандартных переменных можно в функциональном модуле RSVAREXIT\_XXXXXX, где XXXXXX - имя стандартной переменной. В моем примере для 0CML3CM это будет RSVAREXIT\_0CML3CM.

А если нужно сделать свою глобальную переменную, то милости просим в транзакцию CMOD /Управление проектами SAP-расширений/ - расширение RSR00001 BI - Enhancements for Global Variables in Reporting.

## − [Подпрограмма преобразования для 0CALMONTH](https://www.sapforum.pro/forum/viewtopic.php?f=12&t=10878&start=0&sid=8b5955ea199f5c2249ffa43f5dd14f4a)

## Использую в инфопровайдере стандартный инфо-объект 0CALMONTH. У него указана подпрограмма преобразования PERI6, т.е. вводить надо в формате ММ.ГГГГ, и поменять эту подпрограмму я не могу, т.к. этот инфо-объект используется во многих других инфо-провайдерах. А мой заказчик хочет, чтобы при работе с Analyzer, надо было бы вводить значения в формате ГГГГММ.

## − а если попробовать через User-exit? Введённое значение ГГГГММ будет преобразовываться в ММГГГГ и вссё будет кошерно.

## − Я о том же подумал. Вот только как и на что повесить переменную с User-exit-ом? Если на сам 0CALMONTH, то в селекционном экране всё равно нужно будет вводить в виде MM.YYYY.

## − маневр сбоку. Заводим новую левую переменную, делаем её готовой к вводу, а в User-exit-e значение этой переменной трансформируем в нужный формат и присваеваем переменной от 0calmonth, котороя у нас на селекционный экран не выводится.

## − Нашел. Надо поставить соответствующий формат даты в "Ведении собственных постоянных значений пользователя". И тогда PERI6 позволяет вводить в формате ГГГГ.ММ.

## SAP BW variables

## SAP BW variables are parameters of an SAP BW Query - when you run the query, the SAP BW variables are filled with values.

## The SAP BW variable information is obtained using the SAP BW BAPI MDDataProviderBW::GetVariables.

The following types of variables can be created in SAP BW

* Characteristic Value Variables
* Hierarchy Node Variables
* [Text Variables](https://erproof.com/bi/sap-bw-training/sap-bw-text-variable/)
* Formula Variables
* Hierarchy Variables

| Table 1. Selection type property equivalents | |
| --- | --- |
| Value | SAP BW equivalent |
| value | SAP\_VAR\_SEL\_TYPE\_VALUE |
| interval | SAP\_VAR\_SEL\_TYPE\_INTERVAL |
| complex | SAP\_VAR\_SEL\_TYPE\_COMPLEX |
| multiValued | SAP\_VAR\_SEL\_TYPE\_COMPLEX |

### **Text Variables**

These types of variables act as *placeholders for text*. You can use these variables in calculated key figures, structures, and query descriptions. These variables appear as technical name surrounded by ampersands (&). For example

Prior Year Revenue Period = 0062016 -> Prior Year Revenue Period = *&FISCPER\_T01&*

### **Formula Variables**

These types of variables represent numerical values. For instance, you can replace fiscal period with a *variable reference*

Fiscal Period = 0062017 -> Fiscal Period = *FISCPER\_V01*

You can use these variables in formulas for computing conditions and exceptions.

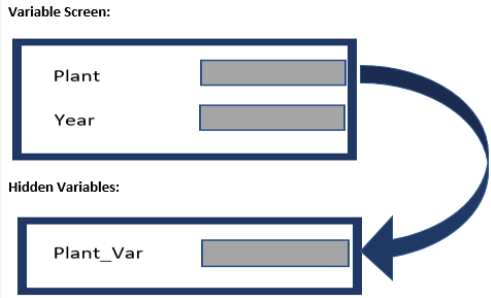
**Processing Types of SAP BW Variables**

The variable processing type governs the way in which a variable gets filled with values during runtime of the query. The following are the various processing types

* Replacement Path
* Customer Exit
* SAP Exit
* User Entry/Default Value
* Authorization

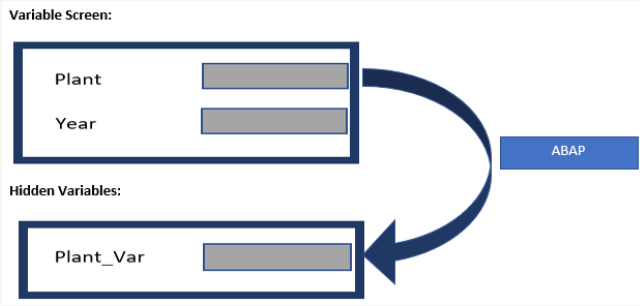
### **Replacement Path**

The replacement path processing type needs to be chosen when *the value of a variable needs to be automatically replaced on execution of the query*. Through the replacement path, we can create hidden variables and later fill them with another variable. For instance, consider a scenario where we have **Plant** and **Year** in the variable screen. **Plant** gets filled from the hidden variable **Plant\_Var**. On the variable screen, you will not see **Plant\_Var**. Based on user selections, the hidden variable is leveraged (используется) to collect the proper data from the database



### **Customer Exit**

Function module **EXIT\_SAPLRRS0\_001** is leveraged via *enhancement* ***RSR00001***. This exit primarily functions with hidden variables but can be written with greater complexity than the replacement path processing type. For instance, if you need to join multiple database tables to identify hidden variables, it will require [ABAP](https://erproof.com/abap/sap-abap-training/) code to derive the logic. Based on user selections, the custom logic is run to collect the proper data from the database



### **SAP Exit**

Function module **RREX\_VARIABLE\_EXIT** is leveraged when the SAP exit processing type is used for a variable. When using this exit, make sure that the variable is filled properly and restrictions are considered in ***FILL\_SP\_1***.

### **Applicable Processing Types by Variable Type**

Please note that not all variable types can leverage all processing types. There are certain restrictions.

Below is the summary table of the applicable processing types for each of the variable types

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Variable Type** | | | | |
| **Processing Type** | Characteristic Value Variables | Hierarchy Node Variables | Text Variables | Formula Variables | Hierarchy Variables |
| User Entry/Default Value | X | X | X | X | X |
| Replacement Path | X |  | X | X |  |
| Customer Exit | X | X | X | X | X |
| SAP Exit | X | X | X | X | X |
| Authorization | X | X |  |  |  |

### Sap blog sap bw creating variable with Processing Type Customer Exit

**Customer Exit Variables**

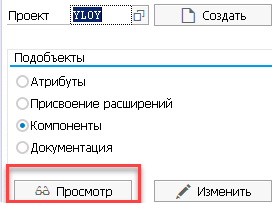
[Содержание](#Содержание)

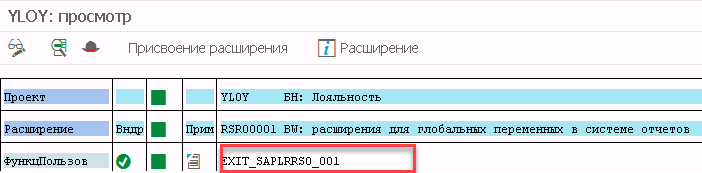
# *from* **Creating Customer Exit Variables with Function Modules**

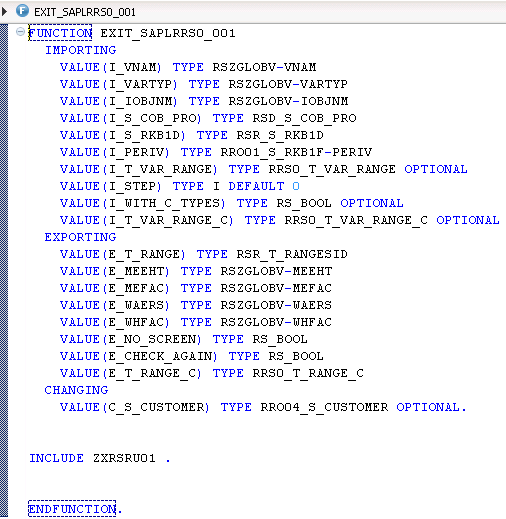
<https://blogs.sap.com/2012/04/07/creating-customer-exit-variables-with-function-modules/>

# **Problem Statement**

Conventionally, as a developer, adding the code for a new customer exit variable is as easy as going to **CMOD**, finding the right project for component EXIT\_SAPLRRS0\_001 and editing the include ZXRSRU01.







But the difficulties start when there are multiple developers and projects working simultaneously. As we all know, the include ZXRSRU01 is a single piece of code that holds logic for all the customer exit variables in the system and this means the following

* Only one developer can work on the code at a given time.
* Transporting the code for a variable means transporting all the code for all variables.
* Work-in-progress variables (переменные незавершенного производства) require removing the unwanted code when a transport is required.
* In large implementations, the include can have several thousand lines of code.

## BW USER EXIT EXPLAINED

<https://zodano.wordpress.com/2009/05/26/bw-user-exit-explained/>

**I\_VNAM** - The variable name.

**i\_t\_var\_range** - contains all the information about the other query variables available in BW.

**l\_s\_range\_low** - is the low limit value of the variable. For non-interval variable, this is the value.

**l\_s\_range\_high** - is the high limit value of the variable. Make sense only for interval type variables.

**l\_s\_range-sign** - denotes whether it’s *‘I’ inclusive*, or *‘E’ exclusive*. Again, make sense for interval type only.

**l\_s\_range-opt** - what type of variable, either *‘EQ’ equal*-type, or ‘*BT’ between-type* (interval).

The SAP BW query exit RSR00001 is *an Enhancement for Global Variables in Reporting*. It is  
called up several times during the execution of a report. The parameter I\_STEP is populated  
with a number from 0 to 3 to specify at what point the enhancement is being called.  
i.e.

* If I\_STEP = 1, then Call has taken place directly *before any variable variable entry has been  
  made*. This can be used to pre populate selection variables.
* If I\_STEP = 2, then then call has taken place *directly after variable entry /* т.е. после нажатия на ОК или на *Проверить/*. This step is only started up *when a variable is not input ready* and could not be filled when I\_STEP was equal to 1.

*Данный шаг вызывается только для тех переменных, которые не поменчены как Готовые для ввода или для манданто зависимых переменных.*

* If I\_STEP = 3, you can *check the values of the variables*. Triggering an exception (RAISE)  
  causes the variable screen to appear once more. After which, I\_STEP=2 is also called again.

*Вызывается после всех остальных шагов.  На данном этапе менять значения переменных нельзя - можно только их проверять.*

* If I\_STEP = 0, then *the enhancement is not called from the variable screen*. The call can come  
  from the authorization check or from the Monitor.

Пример

Using Exit to populate query authorization object dynamically.

First you need to create an authorization object which references a $Variable. In this example I  
am using $ZGMGRANT, which has been linked to the users authorization profile via transaction  
PFCG.

Now within the BW query you need to create and authorization field with the processing type of ‘customer exit’.

# **Customer Exit переменные для Query Designer и вывод сообщений**

<https://helpbw.wordpress.com/2018/12/06/customer-exit-%D0%BF%D0%B5%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D1%8B%D0%B5-%D0%B4%D0%BB%D1%8F-query-designer-%D0%B8-%D0%B2%D1%8B%D0%B2%D0%BE%D0%B4-%D1%81%D0%BE%D0%BE%D0%B1%D1%89%D0%B5%D0%BD%D0%B8/>

При создании запросов в Query Designer возможно создавать Customer Exit переменные, логику работы которых необходимо реализовывать через ABAP.

Для работы с такими переменными используется ФМ EXIT\_SAPLRRS0\_001. В ФМ Include ZXRSRU01, в которым указывается вся реализация Customer Exit переменных.

В ФМ EXIT\_SAPLRRS0\_001существует импортируемая переменная **I\_STEP**, которая может содержать значения 0, 1, 2 или 3.

Рассмотрим пример реализации сообщений на этапе I\_STEP = 3 - когда происходит конечная проверка перед формированием запроса.

Реализацию будем делать следующим образом.

1. В Include ZXRSRU01 будем просто вызывать наш ФМ  *ZEXIT\_VARIABLES*, в котором будет вся логика.
2. В ФМ ZEXIT\_VARIABLES опишем проверку вызываемую на шаге 3.

Рассмотрим код ФМ ZEXIT\_VARIABLES

*ZVARIABLE* - техническое имя Customer exit переменной.

FUNCTION **ZKD1\_EXIT\_VARIABLES**.

*‘CHECK evaluates the subsequent logical expression. If it is true, the processing continues with  
 ‘the next statement.*

CHECK i\_step = 3.

read table i\_t\_var\_range into wa\_loc *‘читаем стандартную таблицу в структуру*

with key *‘ZVARIABLE*

if sy-subrc =0.

    lval = wa\_loc-low.

    hval = wa\_loc-high.

    i\_count = hval - lval + 1.

if i\_count > 10.

      CALL FUNCTION ‘RRMS\_MESSAGE\_HANDLING’

      EXPORTING

            i\_class  =‘класс сообщений’

            i\_type   = ‘E’ *‘тип сообщений*

            i\_number =‘000’ *‘номер сообщения*

            i\_msgv1  =’10’. ‘*текст сообщения для замены*

    RAISE *no\_value*. *‘вызов исключения без текста*

   endif.

endif.

ENDFUNCTION.

CALL FUNCTION  ‘ZEXIT\_VARIABLES’

EXPORTING

      i\_vnam        = i\_vnam

      i\_vartyp      = i\_vartyp

      i\_iobjnm      = i\_iobjnm

      i\_s\_cob\_pro   = i\_s\_cob\_pro

      i\_s\_rkb1d     = i\_s\_rkb1d

      i\_step        = i\_step

      i\_t\_var\_range = i\_t\_var\_range

IMPORTING

      e\_t\_range     = e\_t\_range

EXCEPTIONS

     no\_value = 1.

IF sy—subrc <> 0.

    RAISE no\_value.

ENDIF.

Пример из лояльности

*""$. Region YVAR\_0CALMONTH\_HANA\_EXIT*

WHEN 'YVAR\_0CALMONTH\_HANA\_EXIT'.

CASE i\_step.

WHEN 0. *"authority*

WHEN 1. *"pre popup*

WHEN 2. *"post popup*

READ TABLE i\_t\_var\_range ASSIGNING <fs\_var\_range> *"low limit of YVAR\_0CALMONTH\_BT\_M*

WITH KEY

vnam = 'YVAR\_0CALMONTH\_BT\_M'

iobjnm = '0CALMONTH'.

IF sy-subrc = 0.

CLEAR l\_s\_range.

l\_s\_range-low = <fs\_var\_range>-low.

l\_s\_range-sign = 'I'.

l\_s\_range-opt = 'EQ'.

APPEND l\_s\_range to e\_t\_range.

ENDIF.

WHEN 3. *"check validity*

ENDCASE.

*"$. End Region YVAR\_0CALMONTH\_HANA\_EXIT*

**What is User Exits and Customer Exits in SAP ABAP**

<https://www.guru99.com/what-is-user-and-customer-exits.html>

**Types of Customer Exits**

There are three main types of customer exits

1. Function Module Exits
2. Screen Exits
3. Menu exits

**Function Module Exit**

It  allows customer to add code via a function module at a specific location in an SAP application program.

Syntax - CALL CUSTOMER-FUNCTION '004'

**Screen Exit**

It allows customer to add fields to a screen in an SAP program via a subscreen. The subscreen is called within the standard screen's flow logic.

Format - CALL CUSTOMER-SUBSCREEN CUSTSCR2

**Menu Exit**

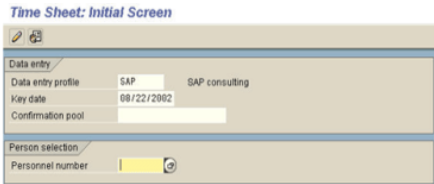
It  allows customer to add items to a pulldown menu in a standard SAP program. These items may be used to call add-on programs or custom screens.

Format: +CUS ( additional item in GUI status )

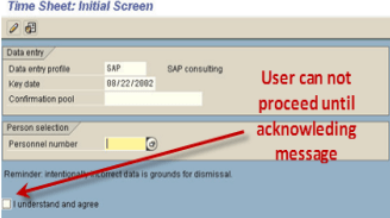
## Examples of Customer Exits

In transaction CAT2 - Time Sheet Entry, HR wishes to include an interactive acknowledgment that knowingly submitting incorrect data is grounds for dismissal.

*Befor*



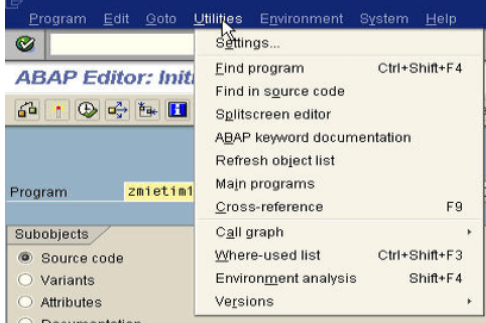
*After*



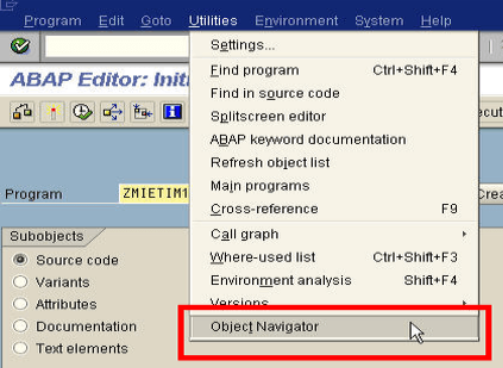
**Example of a Menu Exit**

In transaction SE38 - ABAP Editor, the development team wishes to include a menu link to transaction SE80 - Object Navigator for ease of use.

*Befor*



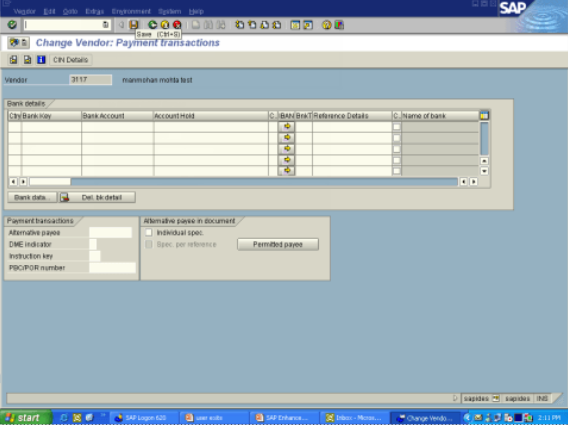
*After*



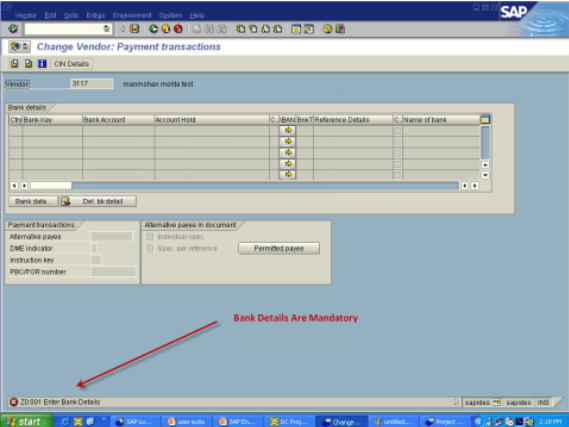
**Example of a Function Module Exit**

The company wants the bank details of the Vendors in the Vendor creation to be mandatory event. So it must flash a error message that 'Please Enter the bank details'

*Befor*

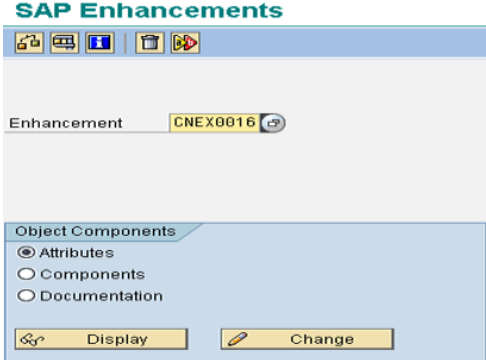


*After*

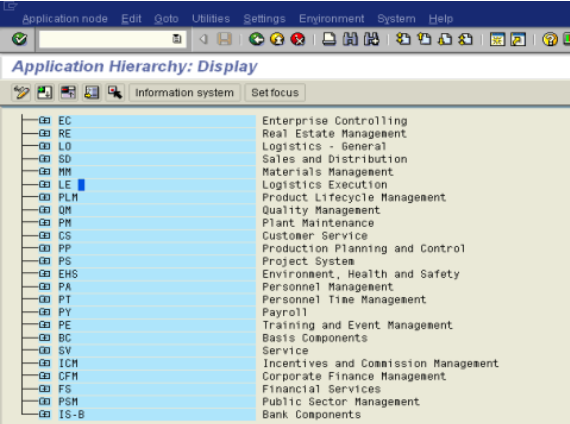


## Locating Customer Exits

In transaction **SMOD** and look into the details-

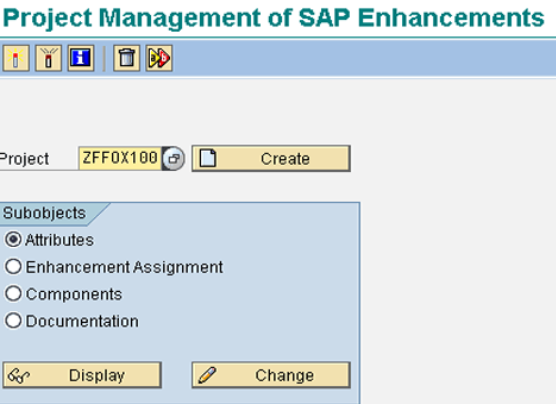


Or in transaction **SE81** you can use the appropriate application area



## ****Create a Customer Exit****

To create a customer exit you first need to create a project in transaction **CMOD**



Later you assign the Customer Exit to your project.

**What is a USER EXIT?**

User Exit serve the same purpose as Customer Exits but they are available only for the **SD** module. The exit is implemented as a call to a Function Module. The code is written by the developer.

Well know User Exit in SD is **MV45AFZZ**

* USEREXIT\_FIELD\_MODIFICATION - To modify screen attributes
* USEREXIT\_SAVE\_DOCUMENT - To perform operations when user hits Save
* USEREXIT\_SAVE\_DOCUMENT\_PREPARE
* USEREXIT\_MOVE\_FIELD\_TO\_VBAK - When user header changes are moved to header work area.
* USEREXIT\_MOVE\_FIELD\_TO\_VBAP - When user item changes are moved to SAP item work area

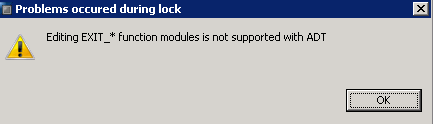
**SAP BI User-Exit RSR00001**

<https://refeline.livejournal.com/8929.html>

Мне понадобилось создать *User-Exit* переменную в *BEx analyzer*. Для этого нужно

1. Создать саму переменную в *BEx Query Designer*, указать Обработку через Customer-Exit.
2. В SAP BI запустить транзакцию **CMOD**.
3. Создать проект и выбрать расширение **RSR00001** /расширение глобальных переменных в отчетах/ и выбрать функцию  **EXIT\_SAPLRRS0\_00**. Но я так понял, что *RSR00001 может использоваться только в одном проекте*, потому попытка повторного использования запрещена ⇒ нужно открывать существующий проект, чтобы добавить туда обработку нашей новой переменной.
4. Кликнуть дважды на функции **EXIT\_SAPLRRS0\_00** - откроется листинг функции. Если проект уже был активирован ранее, то эту функцию можно открыть через транзакцию **se37**.
5. Дважды кликнуть на строке **INCLUDE ZXRSRU01**, чтобы открыть код, который будем редактировать.
6. Редактировать код.

Rem



**BEx with input parameters**

[Содержание](#Содержание)

# **How To Build Dynamic Selections And Filters In SAP BW BEx Queries**

<https://www.saptraininghq.com/how-to-building-dynamic-selections-and-filters-in-sap-bw-bex-queries/>

In order to make reports selections dynamic, OLAP Variables are used, for Characteristics, Texts, Hierarchies and so on.

OLAP Variables are *reusable objects* - any variable created on any Characteristic can be used in any other report on any other Infoprovider that also includes that Characteristic.

To create a new Variable, click the Create New Variable icon clip_image004 and a window appears which contains various fields which are used to define the Variable.

# **Passing Input Parameter from BEx to CV**

[Содержание](#Содержание)

# **Passing Input Parameter from BEx via Composite Provider to Calculation View**

<https://answers.sap.com/questions/720869/passing-input-parameter-from-bex-via-composite-pro.html>