[Содержание](#Содержание)

# [The *previous 12 months* based on the user’s date range [Customer Exit]](#The_previous_12_months)

[I\_T\_VAR\_RANGE  table](#I_T_VAR_RANGE_table)

# [Filter the BRAND field by MERCEDES [BADI]](#Filter_the_BRAND_field_by_MERCEDES)

[BAdi RSROA\_VARIABLES\_HANA\_EXIT\_BADI](#RSROA_VARIABLES_HANA_EXIT)

[Что есть - Enhancement Spot *RSROA\_VARIABLES\_HANA\_EXIT*](#Что_есть)

[Composite Enhancement Spot *ZEAN\_VARIABLE\_HANA\_EXIT*](#Composite_Enhancement_Spot)

[Динамика остатков/приходов](#Динамика_остатков_приходов)

## [Enhancement Implementation](#Enhancement_Implementation)

[Create implementation for RSROA\_VARIABLES\_HANA\_EXIT](#Create_implementation)

Examples

[Calculate from and to values for calendar day based on an input ready calendar day variable](#Ex_1)

['Concatenated: ' || :i\_var\_value\_1 || :i\_var\_value\_2;](#Ex_3)

[from month /07/ to variable /007/](#Ex_5)

EXIT\_SAPLRRS0\_001 Enhancement for Global Variables in Reporting –> ZXRSRU01

# RSROA\_VARIABLES\_EXIT

# *See* Coexistence of BAdI RSROA\_VARIABLES\_EXIT\_BADI and Customer-Exit EXIT\_SAPLRRS0\_001

# <https://blogs.sap.com/2014/08/04/coexistence-of-badi-rsroavariablesexitbadi-and-exit-exitsaplrrs0001/>

# BAdI RSROA\_VARIABLES\_EXIT\_BADI

# <https://blogs.sap.com/2013/07/19/new-badi-rsroavariablesexitbadi-73/>

[Содержание](#Содержание)

# For examination/manipulation of exit variables *previously* only available extension RSR00001 - *Customer Exit Global Variables* [ transaction SMOD] available. The *function block* EXIT\_SAPLRRS0\_001 calls the include ZXRSRU01 which is in the customer namespace.

# *function block* EXIT\_SAPLRRS0\_001 *include* ZXRSRU01

*calls*

As part of the processing process, it is necessary to analyze the values of the *name of the variable* - *I\_VNAM* and the *execution time* - *I\_STEP*. This double nesting quickly leads to confusing and unstructured code. Typical approaches are here

* Using two Nested CASE [*I\_VNAM* and *I\_STEP*] instruction.
* Combination of CASE [*I\_VNAM*] and IF ELSEIF [*I\_STEP*] instructions.
* Outsourcing of source code through the use of includes, function modules, dynamic, dynamic method calls in ABAP OO classes.

# The BAdI RSROA\_VARIABLES\_EXIT\_BADI to fill variables - is the Customer upstream exit [calls the BAdI customer exit on].

# The BAdI is organized in an enhancement spot.

# Последовательность

The standard SAP processing process variables determined by the *GET BADI* command *all active BAdI implementations*.

1. When object *filter the BAdI* uses the info object to which the variable is based that is currently processed.
2. Note
3. The BAdI is called in the I\_STEPs 0, 1 and 2 for each variable. A variable is always based on a feature. In I\_STEP 3 all variables in the current report are available in the form of a table parameter.
4. Then an *instance of each active implementation* generates and invoked with the command CALL BADI method IF\_RSROA\_VARIABLES\_EXIT\_BADI ~ PROCESS of individual implementations.

SAP delivers BAdI definition directly with the default implementation SMOD\_EXIT\_CALL. This implementation is active and has a filter combination IOBJNM = '' OR IOBJNM <> ''.

It is recommend not to mix the use of and BAdI customer exit.

The interface of the BAdI method PROCESS interface is identical to the interface of the function module EXIT\_SAPLRRS0\_001 the customer exit. This means that the source code of the customer exit can be fully adopted.

Note

Migration is not needed - the customer exit persists. The BadI – is an additional concurrent object-oriented way for the processing of exit variables. Advantage of the BAdI variant against the customer exit is better structuring and management of the implementations.

# Вариант 1

METHOD ***if\_rsroa\_variables\_exit\_badi***~***process***.

DATA: l\_s\_range TYPE rsr\_s\_rangesid,

l\_today TYPE umc\_y\_lchaval,

l\_fiscper TYPE umc\_y\_fiscper.

CHECK *l\_vnam* = 'ZTKEFISCPER'.

CHECK *i\_step* = 1.

l\_today = sy-datum.

CALL FUNCTION 'UMC\_CALDAY\_TO\_FISCPER'

EXPORTING i\_periv = 'К4'

i\_calday = l\_today

IMPORTING e\_fiscper = l\_fiscper

EXCEPTIONS invalid = 1

OTHERS = 2.

IF sy-subrc EQ 0.

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

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

l\_s\_range-low = l\_fiscper.

APPEND l\_s\_range TO c\_t\_range.

ENDIF.

# ENDMETHOD.

# Вариант 2

method ***IF\_RSROA\_VARIABLES\_EXIT\_BADI***~***PROCESS***.

DATA: month(6),

fiscper(7),

year (4),

ls\_var\_range TYPE rrs0\_s\_var\_range,

ls\_range TYPE rrrangesid.

CASE *i\_vnam*.

WHEN 'Z\_VS\_SM\_004'.

IF i\_step = 2.

READ TABLE *i\_t\_var\_range* INTO ls\_var\_range WITH KEY vnam = 'Z\_VS\_SM\_001'.

CLEAR ls\_range.

year = ls\_var\_range-low+0(4).

month = ls\_var\_range-low+4(2).

CONCATENATE year '0' month INTO fiscper.

ls\_range-low = fiscper.

ls\_range-sign = 'I'.

ls\_range-opt = ’EQ'. .

APPEND ls\_range TO c\_t\_range.

ENDIF.

ENDCASE.

endmethod.

# RSROA\_VARIABLES\_EXIT

# ZBW\_VAR\_EXIT\_0CALMONTH

# Variable Exit for variable on 0CALMONTH

# CMOD -> function module EXIT\_SAPLRRS0\_001 -> Exit RSR00001.

# With the introduction of SAP BW4/HANA, Enhancement spot is the only option because EXIT\_SAPLRRS0\_001 is no longer supported ⇒ all existing CMOD exits must be converted to Enhancements spots before migrating to BW4/HANA.

# How to create variable exits through standard enhancement spot – RSROA\_VARIABLES\_EXIT

# \_+<https://blogs.sap.com/2014/01/20/how-to-create-variable-exits-through-standard-enhancement-spot-rsroavariablesexit/>

# *I\_STEP = 0* - is called for all variables of processing type *authorization*. This allows for a dynamic approach of identifying the data level authorizations of a given user.

# By using *standard enhancement spot* provided by SAP, we need not populate any logic in the include *ZXRSRU01* and hence can be left blank.

# Since *I\_STEP = 3* is not called by info object, an implementation for all I\_STEP 3 validations can be created with filter value OBJNM = ‘’.

# Usage of SAP HANA Exit Variables for a standard reporting scenario

# \_<https://learntips.net/usage-of-sap-hana-exit-variables-for-a-standard-reporting-scenario-2/>

# Enhancement Spot *RSROA\_VARIABLES\_HANA\_EXIT*.

The objective is to calculate from and to values for calendar day based on an input ready calendar day variable.

Input: 22.08.2015

Result

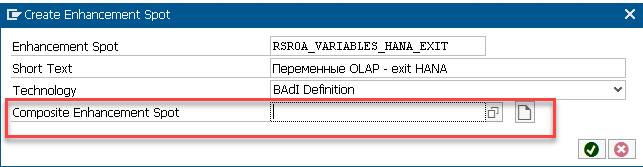
calendar day from 01.01.2015 to 22.08.2015

calendar day from [previous year] 01.01.2014 to [previous year] 22.08.2014

Create Enhancement Spot RSROA\_VARIABLES\_HANA\_EXIT

Start Transaction SE18 -> Create Enhancement Spot *RSROA\_VARIABLES\_HANA\_EXIT*.

? Composite Enhancement Spot



Select “Enh. Spot Element Definitions” Tab and click on the icon “Create BAdI implementation”.



Assign interface for the BAdI:

IF\_RSROA\_VAR\_HANA\_EXIT

Save and activate the interface.

Next, create an Enhancement Implementation, a Composite Enhancement Implementation and a BAdI Implementation.

Enhancement Implementation: ZXX\_CL\_VARIABLE\_HANA

Composite Enhancement Implementation: ZXXXX\_VARIABLE\_HANA\_EXIT

BAdI Implementation: ZXX\_VAR\_HANA\_EXIT\_BADI

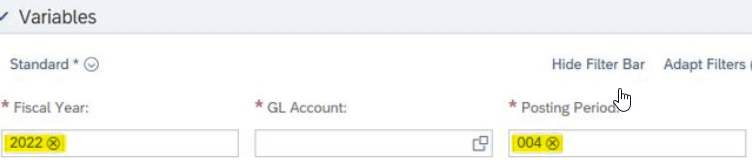
## STEP 3

Implement the class for the YTD variables.

# How to use HANA Exit in SAP Analytics Report [ Selection Screen Default value ]

\_<https://blogs.sap.com/2021/10/11/how-to-use-hana-exit-in-sap-analytics-report-selection-screen-default-value/>

Default values for



CDS View

@AbapCatalog.*sqlViewName*: *'ZCFINRI27NTGRS01'*

@EndUserText.label: 'Network Gross Margin Cube View'

@Analytics: { dataCategory: #CUBE }

@VDM.viewType: *#COMPOSITE*

@AccessControl.authorizationCheck: #CHECK

@AccessControl.personalData.blocking: #BLOCKED\_DATA\_EXCLUDED

@ObjectModel.usageType.sizeCategory: #XXL

@ObjectModel.usageType.dataClass: #MIXED

@ObjectModel.usageType.serviceQuality: #C

@ClientHandling.algorithm: #SESSION\_VARIABLE

@AbapCatalog.buffering.status: #NOT\_ALLOWED

@AbapCatalog.preserveKey: true

@Metadata.ignorePropagatedAnnotations: true

@Metadata.allowExtensions: true

define view ZC\_FIN\_\_RI27\_NETGROSS\_01

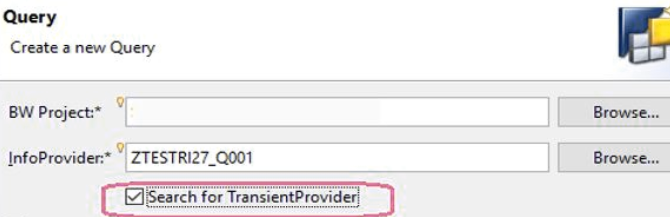
as

select from ZB\_FIN\_RI27\_\_NETGROSS\_01

association [0..\*] to I\_ProfitCenter as ProfitCenter

Query on CDS View

Build the BEx Query with required selection [Fiscal Year / Period] using *CDS SQL name* as **Transients Provider**.

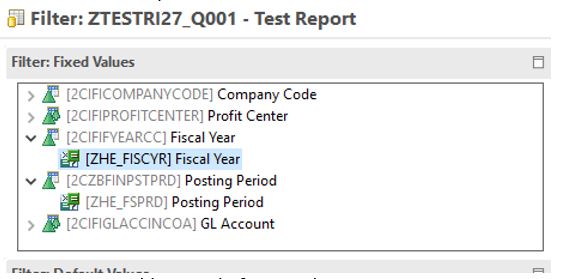


HANA Exit Variable

|  |  |
| --- | --- |
| HANA%20Exit%20for%20FIscal%20Year%20details | Posting%20Period%20HANA%20Exit%20Variable%20details |

BEx

Create respective selection variable for Fiscal Year and Posting Period in BEx with “Processed By” = *HANA Exit*.



BAdI

Enhancement spot - RSROA\_VARIABLES\_HANA\_EXIT

Enhancement Implementation – класс ZXX\_CL\_VARIALBLE\_HANA

METHOD ***if\_rsroa\_var\_hana\_exit***~***process***

BY DATABASE PROCEDURE FOR HDB

LANGUAGE SQLSCRIPT .

c\_value := ' ';

if :i\_vnam = *'ZHE\_FISCYR'* then

if extract ( month from current\_Date ) >= '07' THEN

c\_value := extract ( year from current\_Date ) + 1;

else

c\_value := extract ( year from current\_Date );

end if;

end if;

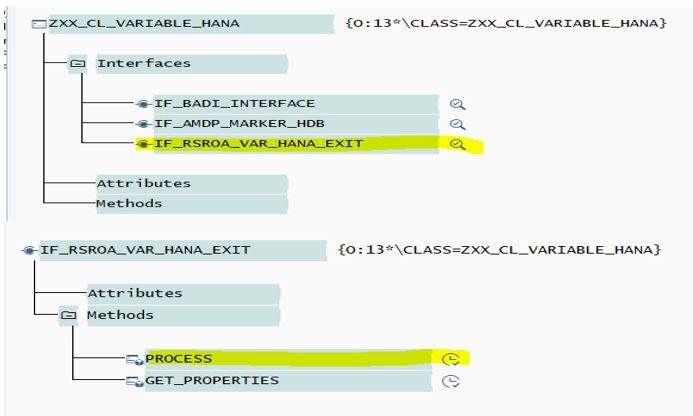
if :i\_vnam = *'ZHE\_FSPRD'* then

if extract ( month from current\_Date ) = ' 01' then c\_value : = '007';

endif;

Test the class method

Open T-code *SE24* and follow the steps as highlighted in **Yellow**



?

RSROA\_VARIABLES\_HANA\_EXIT and RSROA\_VARIABLES\_EXIT – what a difference?

For more details on SAP HANA EXIT see

<https://help.sap.com/docs/SAP_NETWEAVER_750/04030263a0d041309a039fa3ea586720/b77f3073e9d842d6959e03e74dcefa09.html>

# Code Customer Exit Variables with Renewed ABAP

# <https://learntips.net/code-customer-exit-variables-with-renewed-abap/>

[Содержание](#Содержание)

I created a simple BW Query based on SFLIGHT data model to demonstrate renewed ABAP in action. The Query displays *number passengers per airline* who traveled certain distance ranges specified on selection screen.

If Distance Ranges are not in a sequence, then Customer Exit issues an error message

What Customer Exit is doing

* Sets Default Values for Characteristic *Distance Range* Variables;
* Sets Text *Distance Range Text* Variables based on user input;
* Validates Distance Characteristic *Range Variables*.

BADI implementation will be executed for *DISTANCE* InfoObject Variables as well as for *Texts* Variables and at the time of Variables Validation.

Note - if you need to code formulas in the BADI implementation - Filter Val. will be following

Value 1 =  1FORMULA,  Filter  = IOBJNM

Actual coding is done in IF\_RSROA\_VARIABLES\_EXIT\_BADI~PROCESS method of ZCL\_RSROA\_VAR\_EXIT\_DISTANCE class

METHOD ***if\_rsroa\_variables\_exit\_badi***~**process**.

CASE i\_step.

WHEN 1.       *“Before Selection*

CASE i\_vnam.

WHEN ‘DIST\_1’.

c\_t\_range = VALUE #( ( sign = ‘I’ opt = ‘EQ’ low = ‘00600’ ) ).

WHEN ‘DIST\_2’.

c\_t\_range = VALUE #( ( sign = ‘I’ opt = ‘EQ’ low = ‘01000’ ) ).

…  
ENDCASE.

WHEN 2.    *“After selection screen*

CASE i\_vnam.

WHEN ‘DIST\_1H\_TXT’ or ‘DIST\_2H\_TXT’ or ‘DIST\_3H\_TXT’ or ‘DIST\_4H\_TXT’.

c\_t\_range = VALUE #( ( low  =

*replace*( val = |{ i\_t\_var\_range[ vnam = *substring*( val = |{ i\_vnam }| off = 0 len = 6 ) ]–low }| regex = ‘^0+’ with = ” occ = 1 ) ) ).

WHEN ‘DIST\_2L\_TXT’ or ‘DIST\_3L\_TXT’ or ‘DIST\_4L\_TXT’ or ‘DIST\_5L\_TXT’.

c\_t\_range = VALUE #( ( low =

*replace*( val = |{ i\_t\_var\_range[ vnam = i\_vnam+0(5) && |{ i\_vnam+5(1) – 1 }| ]–low + 1 }| regex = ‘^0+’ with = ” occ = 1 ) ) ).

ENDCASE.

WHEN 3.    *“Validation*

TRY.

DO 3 TIMES.

IF i\_t\_var\_range[ vnam = ‘DIST\_’ && |{ sy-index }| ]–low > i\_t\_var\_range[ vnam = ‘DIST\_’ && |{ sy-index + 1 }| ]–low.

DATA(w\_message) = |Range| && |{ sy-index }| && | is greater then Range | && |{ sy-index + 1 }|.

CALL FUNCTION ‘RRMS\_MESSAGE\_HANDLING’

EXPORTING i\_class  = ‘OO’

i\_type   = ‘E’

i\_number = ‘000’

i\_msgv1  = w\_message.

RAISE EXCEPTION TYPE cx\_rs\_error.

ENDIF.

ENDDO.

CATCH cx\_sy\_itab\_line\_not\_found INTO DATA(itab\_line\_not\_found).

ENDTRY.

ENDCASE.

ENDMETHOD.

# The *previous 12 months* based on the user’s date range

[Содержание](#Содержание)

# Customer Exit Variables and Dynamically Restricted Key Figures in BW Reports

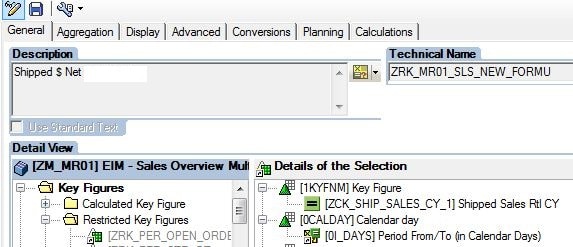
<https://www.dataxstream.com/blog/bw-customer-exit-variables-and-dynamically-restricted-key-figures/>

In a BW Sales Analysis report, user wants to analyze the Sales$ based on the *calendar day range* they entered on the report selection screen and put it in Column A. They would also like to have a separate Column B in the report to view the Sales$ from the *previous 12 months* based on the date range they selected.

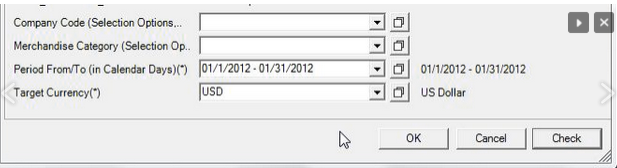
For example - user selected 01/02/2012 – 03/31/2012 as the date range on the selection screen. Then for Column A, the report will display Sales$ from date range 01/02/2012 to 03/31/2012 and for Column B, it will display Sales$ from Calendar Month 01/2011 to 12/2011.

**Step** 1

Create a restricted key figure using key figure *Shipped Sales $* and restrict it by *Calendar day* - 0CALDAY with variable *0I\_DAYS* - *Period From/To*



The BW report selection screen should look like the following

:

# Step 2

# Create a new variable for info object *0CALMONTH* - we will use it to populate the previous 12 months. Make sure to select *Customer Exit* from the Processing By drop down list and choose Selection option from the variable represents drop down.

# 

# Step 3

We need to write the code to populate the previous 12 months for variable ZPRE12\_MONTH\_CDAY.

Run T-Code *CMOD* –> Create a project if none exists –> Components –> EXIT\_SAPLRRS0\_001 /Enhancement for Global Variables in Reporting/ –> ZXRSRU01

# http://www.dataxstream.com/wp-content/uploads/enhancement-screen.jpg

The logic will be as follows

1. Retrieve the date range user  entered on the report selection screen. In this case, the info object is 0CALDAY and variable is *0I\_DAYS*.
2. Calculate the previous 12 month range based on the *date from* value user entered.

WHEN *'ZPRE12\_M0NTH\_CDAY'*

IF I\_STEP = '2'. *"After variable pop-up*

READ TABLE *I\_T\_VAR\_RANGE* INTO WA\_VAR\_RNG

WITH KEY VNAM = '*0I\_DAYS*' IOBJNM = '*0CALDAY*'.

IF SY-SUBRC = 0 .

DAY\_LV = WA\_VAR\_RNG-LOW+6(2). *" день - YYYYMMDD*

MONTH\_LV = WA\_VAR\_RNG-LOW+4(2).

YEAR\_LV = WA\_VAR\_RNG-LOW(4) .

IF MONTH\_LV = '01'.

H\_MONTH\_LV = '12'.

H\_YEAR\_LV = YEAR\_LV - 1.

ELSE.

H\_MONTH\_LV = MONTH\_LV - 1.

H\_YEAR\_LV = YEAR\_LV.

ENDIF.

L\_YEAR\_LV = YEAR\_LV - 1.

CONCATENATE H\_YEAR\_LV H\_MONTH\_LV INTO H\_Month.

CONCATENATE L\_YEAR\_LV MONTH\_LV INTO L\_Month.

L\_S\_RANGE-LOW = L\_Month.

L\_S\_RANGE HIGH = H\_Month.

L\_S\_RANGE-SIGN = 'I'.

L\_S\_RANGE-OPT = 'ВТ'.

APPEND L\_S\_RANGE TO E\_T\_RANGE.

ENDIF.

ENDIF.

# Step 4

# Create another restricted key figure and restrict it by Calendar month and use the variable created - ZPRE12\_MONTH\_CDAY

# 

**I\_T\_VAR\_RANGE table**

[Содержание](#Содержание)

This  table is used to fetch the data given by the end user in the variable whenever we execute the report.

The various fields present in this table are - *LOW*, *HIGH*, *OPT*, *SIGN*

* *OPT* - the various option for the OPT are EQ,BT,NE etc..
* *SIGN* - the various option for the SIGN is I,E.

Now our requirement is to create the MTD report which will displays the output starting of the month till the user input date. Let say the user has given the input date as 20130328 - then MTD report should display the output starting from the range ***20130301*** to ***20130328*.**

**data wa like line of i\_t\_var\_range.**

data : ref\_date type d,

          first\_date type d,

          last\_date type d.

ref\_date = wa-low. *” ref\_date will hold the data 20130328.*

first\_date = ref\_date.

first\_date+6(2) = ’01’. *” In the first date we should get the first date of that month.*

last\_date = ref\_date. *” In the last date we should get the user input date.*

wa-low = first\_date.

wa-high = last\_date.

wa-sign = ‘I’.

wa-opt = ‘BT’.

append wa to e\_t\_range.

# Code Customer Exit Variables with Renewed ABAP

# <https://blogs.sap.com/2016/03/01/code-customer-exit-variables-with-renewed-abap/>

# Filter the BRAND field by MERCEDES

# SAP BI-IP in BW/4HANA part 5. User exit variables.

<https://pavelpetrakov.com/blog/2022/12/09/sap-bi-ip-in-bw-4hana-part-5-user-exit-variables/>

[Содержание](#Содержание)

In this article, we will develop a User exit variable that will filter the BRAND field by MERCEDES.

To process the User Exit variable, it’s necessary to create an *extension implementation* on the class at the RSROA\_VARIABLES\_EXIT extension point, which should use the *IF\_RSROA\_VARIABLES\_EXIT\_BADI* interface.

In the class, you need to redefine the PROCESS method, in which you need to fill in the C\_T\_RANGE table.

### Step 1 - Create a variable ZZE0001\_VAR01 for the BRAND info object

# Enter the name and description of the variable, select the type – *Customer Exit*

# https://pavelpetrakov.com/wp-content/uploads/2022/12/bi_ip_part_5_step_1_1_set_up_var01.png

# Remove *Input-Ready* and save the variable

# 

### Step 2 - Add a variable to the BEx query ZZLICAR01\_QP01

### Add a variable to the BEx query ZZLICAR01\_QP01

Open the Filter tab in the BEx query\right-click and select *Add Characteristics* - select the BRAND attribute and add it.

# Click on the BRAND attribute with the right mouse button and select *Restrict*.

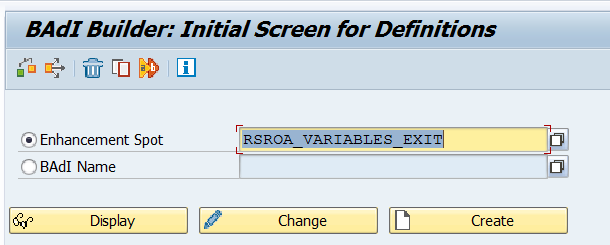
# In the window that opens, go to the Variables tab and move the variable ZZE0001\_VAR01 to the right side of the window, then click OK.

### Step 3 - Check if there is an implementation of the RSROA\_VARIABLES\_EXIT extension

At this step, it’s important to check whether the *implementation* of the RSROA\_VARIABLES\_EXIT extension already exists in the system. This is necessary because if you make multiple implementations of the same extension - the system will look for variable handling in each extension implementation, which will lead to confusion.

THE BEST PRACTICE is to have one extension implementation.

You need to log into transaction SE18 and enter the value RSROA\_VARIABLES\_EXIT in the Enhancement Spot field.



In the window that appears, let’s check if there is already a *Z\** implementation /ZBI\_RSROA\_VARIABLES\_EXIT/ of the extension (there is one in the screenshot below).

# 

# 

# In this case, it is necessary to analyze the existing implementation – most likely it determines how the ABAP class should be called so that there would be permanent processing. The screenshots below show that this implementation will read all classes with the name “ZCL\_VAR\_” + the name of the variable, i.e. in our case ZCL\_VAR\_ZZE0001\_VAR01

# 

METHOD **if\_rsroa\_variables\_exit\_badi~process**.

DATA:

l\_v\_classname TYPE seoclsname,

l\_o\_object TYPE REF TO object,

l\_ce\_method TYPE string VALUE 'IF\_RSROA\_VARIABLES\_EXIT\_BADI~PROCESS',

l\_s\_var\_range LIKE LINE OF i\_t\_var\_range,

l\_method\_exists TYPE rs\_bool.

IF i\_step < 3.

*' Собираем имя класса по маске ZBWCL\_VAR\_[Имя переменной]*

CONCATENATE 'ZCL\_VAR\_' i\_vnam INTO l\_v\_classname.

ELSEIF i\_step = 3.

*' Для i\_step = 3 отдельный класс. При необхдимости реализовать разные логики*

*' для отчетности и бюджетирования анализировать i\_rkbld-infocube или i\_rkbld-compid и*

*' создать 2 разных класса*

l\_v\_classname = 'ZCL\_VAR\_ISTEP3'.

ENDIF.

IF l\_v\_classname <> ' '.

TRY.

CREATE OBJECT l\_o\_object TYPE (l\_v\_classname).

CATCH cx\_class\_not\_existent.

ENDTRY.

TRY.

CALL METHOD l\_o\_object->(l\_ce\_method)

EXPORTING

i\_vnam = i\_vnam

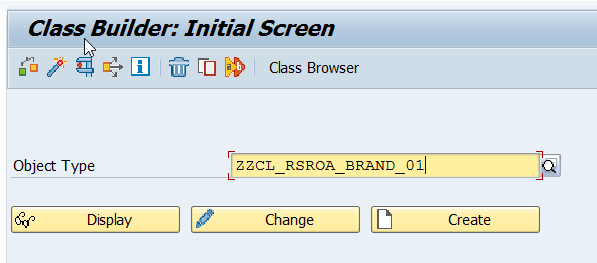
i\_vartyp = i\_vartyp

i\_iobjnm = i\_iobjnm

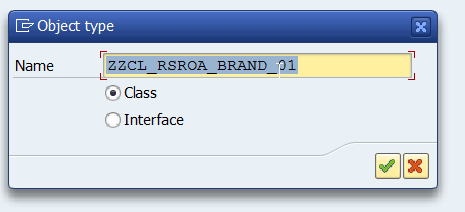
### Step 4 - Create an ABAP class to handle the variable.

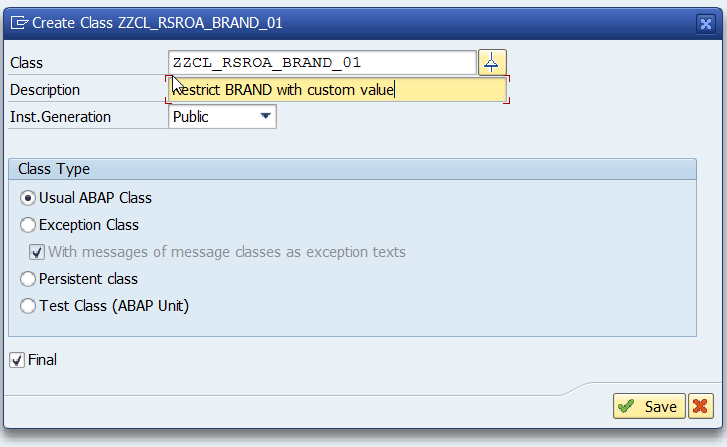
Let’s create our own class /I specifically call it ZZCL\_RSROA\_BRAND\_01, not ZCL\_VAR\_ZZE0001\_VAR01, so that in the next step we can create our own extension implementation to show how it is done. But on combat projects it is necessary to use an already created *extension implementation*/.

Open transaction SE24, enter the name of the class and click *Create*



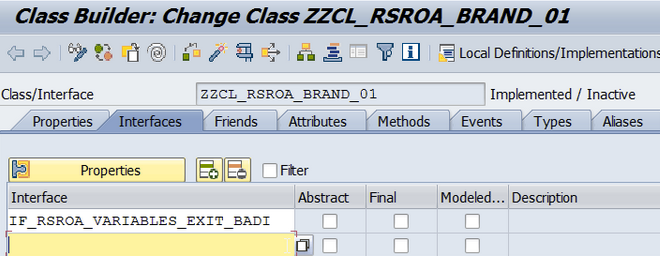
In the windows that appear, we indicate that this is a class, not an interface, and specify its description



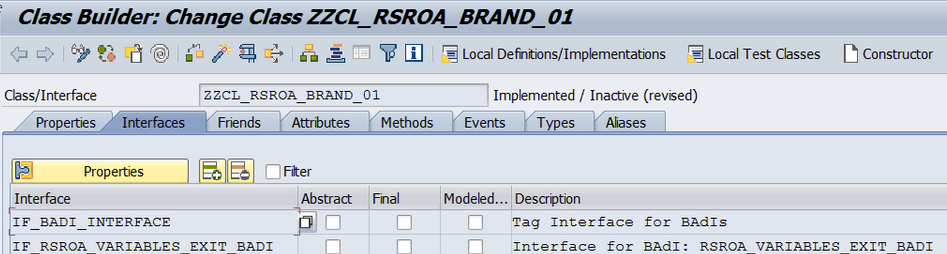


Specify what we will save in the temp package.

In the window that appears, go to the *Interfaces* tab and enter IF\_RSROA\_VARIABLES\_EXIT\_BADI, press enter.



Система автоматически подтянет еще один интерфейс **IF\_BADI\_INTERFACE**, необходимый для работы с BADI.



Enter the text as in the screenshot below

method ***IF\_RSROA\_VARIABLES\_EXIT\_BADI~PROCESS***.

DATA l\_s\_range LIKE LINE OF *c\_t\_range*.

IF I\_STEP = 1.

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

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

l\_s\_range-LOW = 'MERCEDES'.

APPEND l\_s\_range to *c\_t\_range*.

ENDIF.

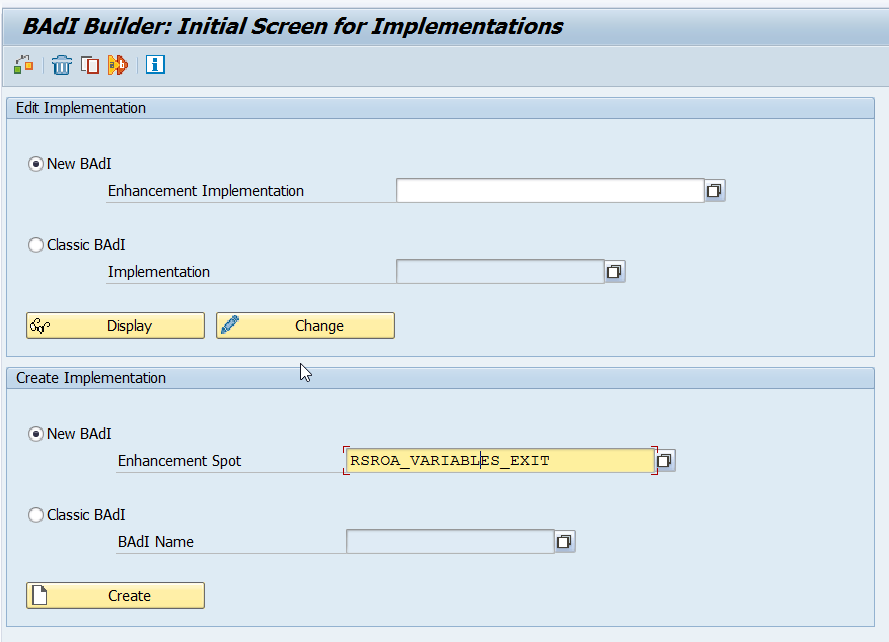
endmethod.

Save and activate

### Step 5 - Creating an implementation of the RSROA\_VARIABLES\_EXIT extension point extension

Once again, it is highly discouraged to create your own extension implementation if it already exists in the system. At this step, we are creating it exclusively for educational purposes.

Go into transaction SE19 and enter RSROA\_VARIABLES\_EXIT and click Create.



We introduce a description of the extension implementation.



~~ZZSANDBOX\_RSROA\_VARIABLES\_EXIT~~

~~SANDBOX Exit variables~~

ZZCUSTOM\_RSROA\_VARIABLES\_EXIT

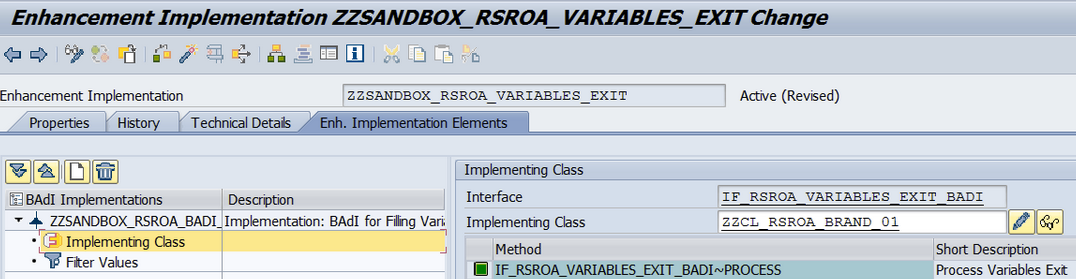
Custom Exit variables

Specify the package.

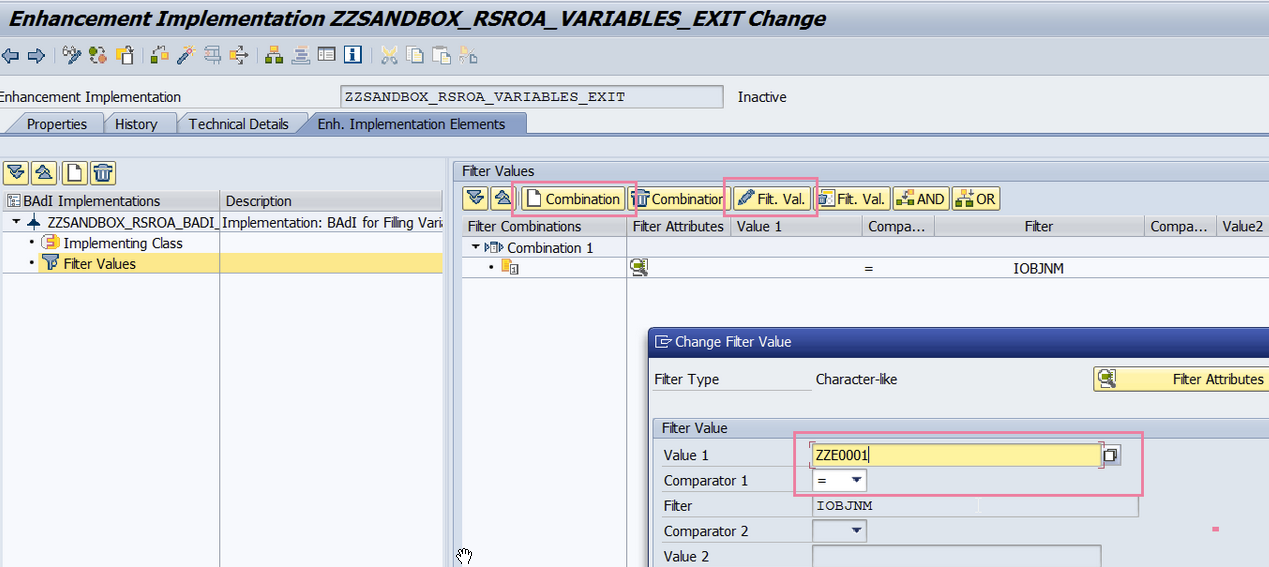
Now we need to specify on which class we are implementing the extension. We can close this window without filling it. We will return to it later.

Click on create.

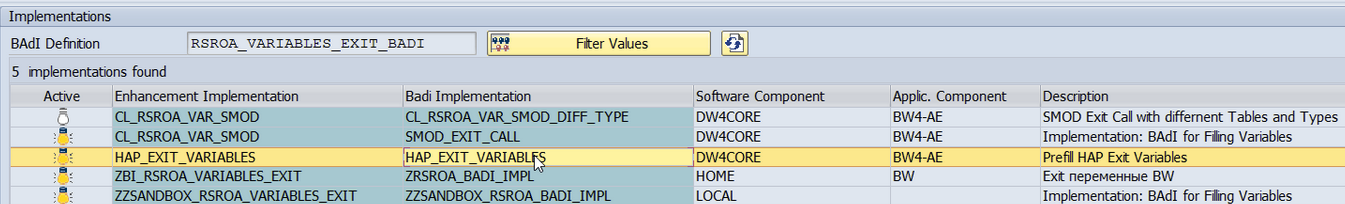
It should turn out like this



Immediately add a filter that will say that this implementation will only process our variable.

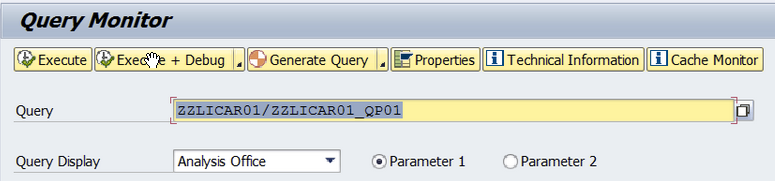


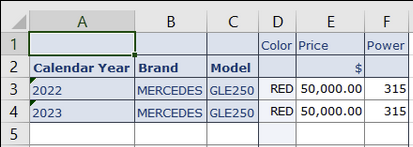
In my case, in transaction se18, we see 2 implementations of the extension – this is bad, I will then delete the one that was created in this step [in order to show how to do it].



### Check the result

Open the RSRT transaction, enter our BEx request and click Execute.





# possible badi instead of cmod for bex variables?

# <https://answers.sap.com/questions/12060348/possible-badi-instead-of-cmod-for-bex-variables.html>

We want use badi enhancement for bex variables istead of customer exit.

We used RSROA\_VARIABLES\_EXIT for this and created /FITGL/VAR\_EXIT\_GUEBG implemantation.

We use this badi for /FITGL/GUEBG infoobject.

Just simple query for to get current date to ZENTRYDATE variable.

Here is the code;

METHOD if\_rsroa\_variables\_exit\_badi~process.  
\* c\_s\_customer-vproctp = 1.  
  
CASE i\_vnam.  
  
WHEN 'ZENTRYDATE'.  
  
DATA l\_s\_range TYPE rrrangesid.  
  
IF i\_step = 1.  
l\_s\_range-low = sy-datum.  
l\_s\_range-sign = 'I'.  
l\_s\_range-opt = 'EQ'.  
APPEND l\_s\_range TO c\_t\_range.  
  
  
ENDIF.  
  
ENDCASE.  
  
  
ENDMETHOD.

But code doesnt work..we debugged code and noticed that,code goes to ZXRSRU01 at the end.but nothing in there and ZENTRYDATE becomes null.

is there a way to using just badi for bex variables.

**−** The code is simple and perfect. Just wanted to add few things to your activities.

1) In the include of EXIT\_SAPLRRS0\_001 of the enhancement project in CMOD, we need to add a code to call the BADI. Hope you have done that.

2) The Append statement l\_s\_range to c\_t\_range needs to be checked once. Just check for the proper naming convention. Sometimes, a small mistake will make the code not to work. for example the name might be e\_t\_range.

3) Please check the format of ZENTRYDATE. It should be same as sy-datum. Else the result will be blank

Check the above points and if it doesn't help, let us know. We will analyse further then.

**−** The Customer-exit Variable framework has changed as of BW release 7.3. The new BAdI RSROA\_VARIABLES\_EXIT\_BADI is called in Function Module RRS\_VAR\_EXIT.

I suggest to debug in Function Module RRS\_VAR\_EXIT. Put a break-point here and then look how the customer-exit variables are processed.

It might be that one of BAdI implementations still calls the legacy Enhancement. This could be an explanation that here a clearing of the variable takes place.

For testing purposes you should consider to create a simplified BEx Query to avoid that it is becoming too time-consuming.

Please refer to the following blogs for more information:

* [New BAdI RSROA\_VARIABLES\_EXIT\_BADI (7.3)](https://answers.sap.com/community/business-explorer/blog/2013/07/19/new-badi-rsroavariablesexitbadi-73);
* [Coexistence of BAdI RSROA\_VARIABLES\_EXIT\_BADI and Customer-Exit EXIT\_SAPLRRS0\_001](https://answers.sap.com/community/business-explorer/blog/2014/08/04/coexistence-of-badi-rsroavariablesexitbadi-and-exit-exitsaplrrs0001).

Last but not least, you can also have a look at my blog [Framework for Customer Exit OLAP Variables](https://answers.sap.com/community/business-explorer/blog/2014/12/22/framework-for-customer-exit-olap-variables) for an alternative approach.

# Установите обычную вентиляционную трубу для стиральной машины. Воздух, попавший в сливную трубу стиральной машины, должен куда-то уходить, и когда он пытается вернуться обратно в сливную трубу, может возникнуть воздушная пробка.

# Установите воздуховыпускную трубу (AAP), если устанавливать обычную вентиляционную трубу неудобно или слишком дорого. Обратитесь к сантехническим кодам в вашем районе, чтобы узнать, разрешено ли это компромиссное решение.

# 

# Мини сифон для стиральной машины

# 

# How to Use Key Date Derivation Type

<https://visualbi.com/blogs/sap/sap-bw-hana/sap-bw4hana/sap-bw-4hana-version-2-0-use-key-date-derivation-type/>

### **Generating master data for Time Characteristics**

<https://blog.maruskin.eu/2022/03/generating-master-data-for-time.html#google_vignette>

# Simplify your BW enhancements for global reporting variables

<https://blogs.sap.com/2007/01/16/simplify-your-bw-enhancements-for-global-reporting-variables/>

### SAP BW Reporting Variables:Customer Exit

<https://expertisesapbi.blogspot.com/2011/06/blog-post.html>

# 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\_001**существует импортируемая переменная I\_STEP, которая может содержать значения 0, 1, 2 или 3.

I\_STEP = 0 — вызывается, если в запросе есть переменные, работающие с объектами авторизации.

I\_STEP = 1 — вызывается для всех Customer Exit переменных на этапе, до того как появится селекционный экран. На этом шаге можно устанавливать значения переменных, определять их и т.п.

I\_STEP = 2 — вызывается не для всех Customer Exit переменных. Вызывается после процесса установки переменных. Т.е. после нажатия на ОК или на «Проверить». Данный шаг вызывается только для тех переменных, которые не поменчены как «Готовые для ввода» или для манданто зависимых переменных.

I\_STEP = 3 — вызывается после всех остальных шагов.  На данном этапе менять значения переменных нельзя, можно только их проверять.

Рассмотрим пример реализации сообщений на этапе I\_STEP = 3 , т.е. 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, ниже будут некоторые пояснения.

ZVARIABLE — техническое имя Customer exit переменной.

FUNCTION ZKD1\_EXIT\_VARIABLES.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*тут идёт какой-то код и логика

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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.

Реализуем свою логику, вызываем ФМ для отображения сообщений  ‘RRMS\_MESSAGE\_HANDLING’.

Указываем исключение RAISE no\_value. Исключение no\_value необходимо указать в ФМ на вкладке «Особые ситуации».

Далее возвращаемся в Include ZXRSRU01 и реализовываем ту часть когда, которая вызывает наш ФМ.

Передаём стандартные параметры в ФМ, и обязательно указываем при экспорте наше исключение no\_value.

И далее, если sy—subrc не равно 0, т.е. есть исключения, указываем RAISE no\_value. Это позволит не разрешать формировать отчё

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.

Тут ещё немного написано про вывод сообщений

[Displaying messages in BEx with i\_step = 3](https://archive.sap.com/discussions/thread/3818221)

[Customer Exit Variables in SAP BI](https://wiki.scn.sap.com/wiki/display/BI/Customer+Exit+Variables+in+SAP+BI) — на Вики SAPа есть информация про Customer Exit, в том числе тип Formula.

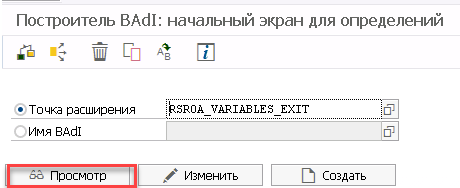
**BAdi RSOA\_VARIABLES\_EXIT\_BADI**

### Step 3 - Check if there is an implementation of the *RSROA\_VARIABLES\_EXIT* extension

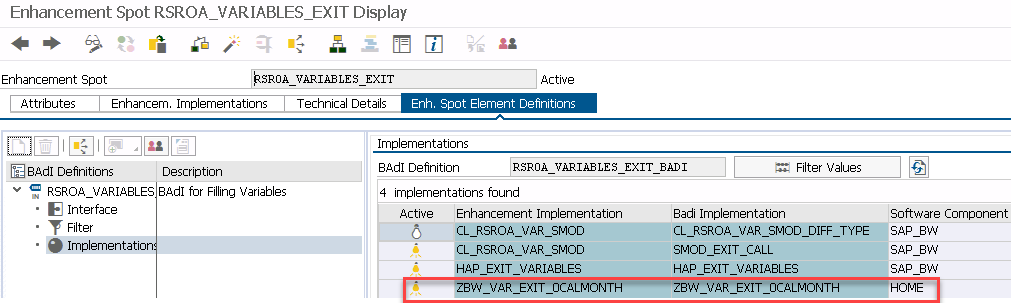
It’s important to check whether the implementation of the RSROA\_VARIABLES\_EXIT extension already exists in the system. This is necessary because if you make multiple implementations of the same extension, the system will look for variable handling in each extension implementation, which will lead to confusion.

THE BEST PRACTICE is to have one extension implementation.

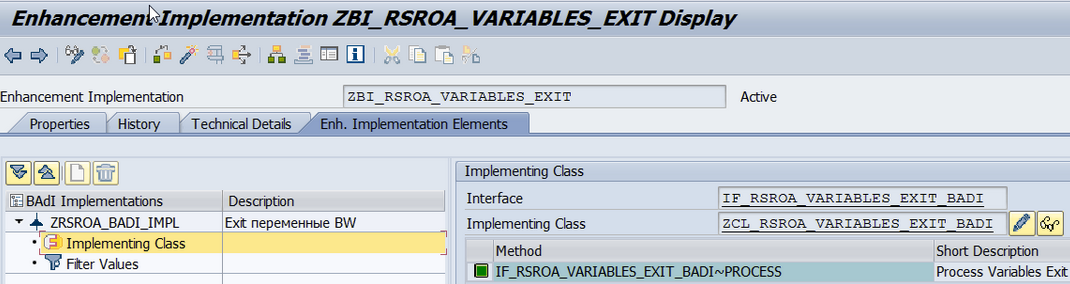
You need to log into transaction ***SE18*** and enter the value RSROA\_VARIABLES\_EXIT in the Enhancement Spot field.

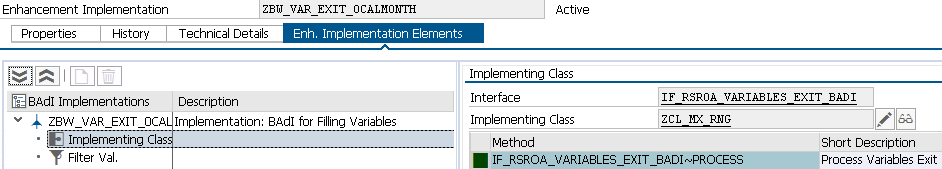


In the window that appears, let’s check if there is already a Z\* implementation of the extension.



In this case, it is necessary to analyze the existing implementation – most likely it determines how the ABAP class should be called so that there would be permanent processing. The screenshots below show that this implementation will read all classes with the name “*ZCL\_VAR\_”* + the name of the variable, i.e. in our case ZCL\_*VAR\_ZZE0001\_VAR01*





method ***IF\_RSROA\_VARIABLES\_EXIT\_BADI~PROCESS***.

*\* data wa like line of c\_t\_range.*

*\* data ref\_date type d.*

*\* ref\_date = wa-high.*

*\* wa-sign = 'I'.*

*\* wa-opt = 'EQ'.*

*\* append wa to c\_t\_range.\*/*

data: month(2),

l\_max\_range(6),

year(4),

ls\_var\_range TYPE rrs0\_s\_var\_range,

ls\_range TYPE rrrangesid.

CASE *i\_vnam*.

WHEN 'Z\_CUSTOM\_RANGE'.

IF i\_step = 2.

READ TABLE i\_t\_var\_range INTO ls\_var\_range WITH KEY vnam = 'Z\_CUSTOM\_RANGE'.

CLEAR ls\_range.

year = ls\_var\_range-high+0(4).

month = ls\_var\_range-high+4(2).

CONCATENATE year month INTO l\_max\_range.

ls\_range-low = l\_max\_range.

ls\_range-sign = 'I'.

ls\_range-opt = 'EQ'.

APPEND ls\_range TO c\_t\_range.

ENDIF.

ENDCASE.

endmethod.

## **Customer Exit Variable code For Last month & For last 2 Months based on the Date interval( From -To)**

<https://community.sap.com/t5/technology-q-a/customer-exit-variable-code-for-last-month-for-last-2-months-based-on-the/qaq-p/10756644>

**BAdi RSROA\_VARIABLES\_HANA\_EXIT\_BADI**

[Содержание](#Содержание)

**Что есть** - Enhancement Spot *RSROA\_VARIABLES\_HANA\_EXIT*

[Содержание](#Содержание)

Enhancement Spot - *RSROA\_VARIABLES\_HANA\_EXIT*

Enhancement Method - *Object Plug-in (BAdI)*

Description - *OLAP variables HANA exit*

Package - *RSROA\_VAR*

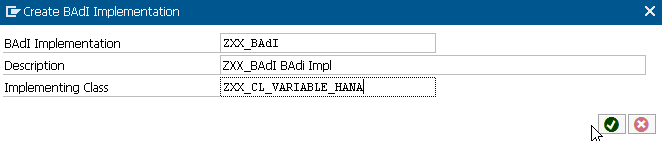
Interface - *IF\_RSROA\_VAR\_HANA\_EXIT*

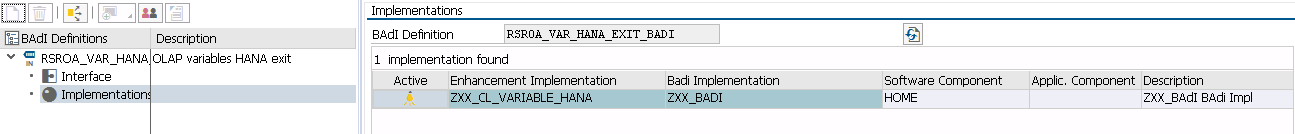
**Composite Enhancement Spot**

[Содержание](#Содержание)

Composite Enhancement Spot - ZXX\_CL\_HANA

Composite Enh. Implementation





Динамика остатков/приходов

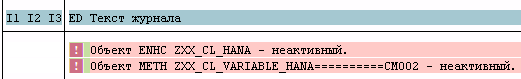
[Содержание](#Содержание)

*Enhancement Implementation* - ZXX\_CL\_VARIABLE\_HANA

*BAdI implementation* - ZXX\_BADI /Description - ZXX\_BAdI BAdi Impl/

*Interface* - IF\_RSROA\_VAR\_HANA\_EXIT

*Implementing Class* - ZXX\_CL\_VARIABLE\_HANA



Объект ENHC ZXX\_CL\_HANA - неактивный.

Объект METH ZXX\_CL\_VARIABLE\_HANA==========CM002 - неактивный.

Объект ENHO ZPPD\_CL\_VARIABLE\_HANA - неактивный.

*где*

ZXX\_CL\_HANA - Composite Enhancement Spot

Транзакция для ведения Composite Enhancement Spot

Transaction code for *HANA EXIT Enhancement Implementation management*

***se19*** *-* 

***se20*** - 

# See

# Creating, Editing, and Deleting Enhancement Spots

<https://help.sap.com/docs/SAP_NETWEAVER_701/6f40b8f46c4b10149770d29604c9a4eb/3b0a39426f79f83ae10000000a1550b0.html?version=7.01.24>

## **Enhancement Implementation**

[Содержание](#Содержание)

The BAdI implementation class requires an AMDP class containing the following interfaces

* IF\_BADI\_INTERFACE
* IF\_AMDP\_MARKER\_HDB
* IF\_RSROA\_VAR\_HANA\_EXIT.

As an example, you can use class *CL\_RSROA\_HANA\_EXIT\_FALLBACK*.

Example 1

CLASS **cl\_rsroa\_hana\_exit\_fallback** *DEFINITION*

PUBLIC

FINAL

CREATE PUBLIC .

PUBLIC SECTION.

*INTERFACES if\_badi\_interface .*

*INTERFACES if\_amdp\_marker\_hdb .*

*INTERFACES if\_rsroa\_var\_hana\_exit .*

PROTECTED SECTION.

PRIVATE SECTION.

ENDCLASS.

CLASS **CL\_RSROA\_HANA\_EXIT\_FALLBACK** *IMPLEMENTATION*.

METHOD ***if\_rsroa\_var\_hana\_exit~get\_properties***

by database procedure for hdb language SQLSCRIPT.

*/\* C\_IS\_ACTIVE should have the value 'X' to determine that it will contain the value of a variable.*

*The system can contain no more than one BAdI implementation with the value 'X for the query/variable name combination.*

*\*/*

*c\_is\_active* := 'X';

IF :i\_vnam = *'HANA\_EXIT\_FORMULA*' THEN

*/\* C\_TS\_VNAM\_INDEX defines the allocation of variable values from the query to the I\_VAR\_VALUE\_X parameters of method if\_rsroa\_var\_hana\_exit~process.*

*The specification of row {VNAM\_COUNTRY, 1} in parameter C\_TS\_VNAM\_INDEX stipulates that the value of variable VNAM\_COUNTRY is assigned to parameter I\_VAR\_VALUE\_1.*

*A maximum of 20 entries of this type can be made.*

*\*/*

*c\_ts\_vnam\_index* = select 'ZFK\_LAND' as vnam, 1 as index from dummy

union

select 'other\_var' as vnam, 2 as index from dummy;

END IF;

ENDMETHOD.

METHOD ***if\_rsroa\_var\_hana\_exit~process***

by database procedure for hdb language SQLSCRIPT.

*// c\_value should specify the value of the variable.*

c\_value := '';

IF :i\_vnam = '*HANA\_EXIT\_FORMULA*' THEN

IF :i\_var\_value\_1 = 'DE' THEN

c\_value := 2;

ELSE

c\_value := 5;

END IF;

END IF;

IF :i\_vnam = '*HANA\_EXIT\_0SOURSYSTEM*' THEN

c\_value := '';

END IF;

ENDMETHOD.

ENDCLASS.

Example 2

CLASS **ZCL\_VAR\_HANA\_EXIT** *IMPLEMENTATION*.

METHOD ***IF\_RSROA\_VAR\_HANA\_EXIT~GET\_PROPERTIES***

by database procedure for hdb language SQLSCRIPT.

IF :I\_VNAM = *'ZJB\_CONCAT*' THEN

c\_is\_active := 'X';

c\_ts\_vnam\_index = select '*ZJB\_VAL1*' as vnam, 1 as index from dummy

union

select '*ZJB\_VAL2*' as vnam, 2 as index from dummy;

END IF;

ENDMETHOD.

METHOD ***IF\_RSROA\_VAR\_HANA\_EXIT~PROCESS***

by database procedure for hdb language SQLSCRIPT.

IF :i\_vnam = '*ZJB\_CONCAT*' THEN

c\_value := 'Concatenated: ' || *:i\_var\_value\_1* || *:i\_var\_value\_2*;

END IF;

ENDMETHOD.

ENDCLASS.

<https://www.se80.co.uk/sap-oop/?class=if_rsroa_var_hana_exit&method=process>

# IF\_RSROA\_VAR\_HANA\_EXIT OO Class - Inferface for BAdi: RSROA\_VARIABLES\_HANA\_EXIT

# IF\_RSROA\_VAR\_HANA\_EXIT is a standard [SAP object class](https://www.se80.co.uk/training-education/sap-oop-abap/) available within R/3 SAP systems depending on your version and release level. You can view/maintain the class details by entering its name into the relevant SAP transactions such as SE24, SE80 or even SE84.

## Method list of IF\_RSROA\_VAR\_HANA\_EXIT SAP class

# the implementation of a method is similar to a function module and can be called in a very similar way using CALL METHOD. Methods are referenced using the following syntax: CALL METHOD =>methodname EXPORTING/IMPORTING...

## Parameters of Method GET\_PROPERTIES

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Data Type** | **Description** | **Default Value** |
| **I\_COMPID** | Importing | **TYPE** RSZCOMPID | Name (ID) of a reporting component |  |
| **I\_INFOPROV** | Importing | **TYPE** RSINFOPROV | InfoProvider |  |
| **I\_IOBJNM** | Importing | **TYPE** RSIOBJNM | InfoObject |  |
| **I\_VARTYP** | Importing | **TYPE** RSZVARTYP | Type of a Report Variable |  |
| **I\_VNAM** | Importing | **TYPE** RSZVNAM | Name (ID) of a Report Variable |  |
| **C\_IS\_ACTIVE** | Changing | **TYPE** RS\_BOOL | Is this implementation active for this compid/vnam |  |
| **C\_TS\_VNAM\_INDEX** | Changing | **TYPE** NT\_TS\_VNAM\_INDEX | Allocation of variable values to i\_var\_value parameters |  |

## Exceptions of Method GET\_PROPERTIES

This method does not have any exceptions

## Example ABAP coding

DATA: lo\_class TYPE REF TO IF\_RSROA\_VAR\_HANA\_EXIT.  
DATA: lv\_C\_IS\_ACTIVE TYPE RS\_BOOL,  
lv\_C\_TS\_VNAM\_INDEX TYPE NT\_TS\_VNAM\_INDEX,  
lv\_I\_COMPID TYPE RSZCOMPID,  
lv\_I\_INFOPROV TYPE RSINFOPROV,  
lv\_I\_IOBJNM TYPE RSIOBJNM,  
lv\_I\_VARTYP TYPE RSZVARTYP,  
lv\_I\_VNAM TYPE RSZVNAM,  
lv\_other TYPE c.  
  
CALL METHOD lo\_class=>GET\_PROPERTIES(  
EXPORTING   
I\_COMPID = lv\_I\_COMPID   
I\_INFOPROV = lv\_I\_INFOPROV   
I\_IOBJNM = lv\_I\_IOBJNM   
I\_VARTYP = lv\_I\_VARTYP   
I\_VNAM = lv\_I\_VNAM  
CHANGING  
C\_IS\_ACTIVE = lv\_C\_IS\_ACTIVE  
C\_TS\_VNAM\_INDEX = lv\_C\_TS\_VNAM\_INDEX ).

**Create implementation for RSROA\_VARIABLES\_HANA\_EXIT**

[Содержание](#Содержание)

Select “Enh. Spot Element Definitions” Tab and click on the icon “Create BAdI implementation”.

Assign interface for the BAdI:

IF\_RSROA\_VAR\_HANA\_EXIT

Save and activate the interface.

Next, create an Enhancement Implementation, a Composite Enhancement Implementation and a BAdI Implementation.

Enhancement Implementation *ZXX\_CL\_VARIABLE\_HANA*

Composite Enhancement Implementation *ZXXXX\_VARIABLE\_HANA\_EXIT*

BAdI Implementation *ZXX\_VAR\_HANA\_EXIT\_BADI*

Enhancement implementation - *ZEAN\_CL\_VARIABLE\_HANA*

Short Text - *OLAP variables HANA exit impl*

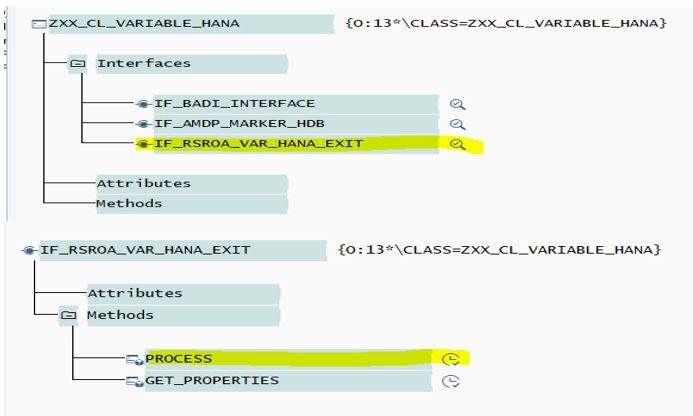
Composite Enhancement implementation - *ZEAN\_CL\_VARIABLE\_HANA\_EXIT*

Enhancement spot - RSROA\_VARIABLES\_HANA\_EXIT

Enhancement Implementation – класс ZXX\_CL\_VARIALBLE\_HANA

Test the class method

Open T-code *SE24* and follow the steps as highlighted in **Yellow**



?

RSROA\_VARIABLES\_HANA\_EXIT and RSROA\_VARIABLES\_EXIT – what a difference?

For more details on SAP HANA EXIT see

<https://help.sap.com/docs/SAP_NETWEAVER_750/04030263a0d041309a039fa3ea586720/b77f3073e9d842d6959e03e74dcefa09.html>

RSROA\_VARIABLES\_HANA\_EXIT for period variable sap blog

# Usage of SAP HANA Exit Variables for a standard reporting scenari

<https://learntips.net/usage-of-sap-hana-exit-variables-for-a-standard-reporting-scenario/>

# AMDP based BEx Customer HANA Exit

<https://learntips.net/amdp-based-bex-customer-hana-exit/>

# 

# 

**Examples**

# Usage of SAP HANA Exit Variables for a standard reporting scenario

# <https://learntips.net/usage-of-sap-hana-exit-variables-for-a-standard-reporting-scenario/>

[Содержание](#Содержание)

The objective is to calculate from and to values for calendar day based on an input ready calendar day variable.

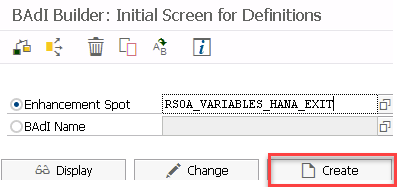
Input - 22.08.2015

Result

from current year - 01.01.2015; to current year - 22.08.2015

from previous year - 01.01.2014; to previous year - 22.08.2014

## ***Step 1*** - *Create the* *enhancement spot* based on BAdI RSOA\_VARIABLES\_HANA\_EXIT

***SE18*** -> .

## ***Step 2*** - *Create BAdI implementation*

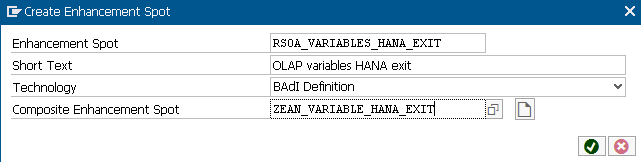
Select *Enh. Spot Element Definitions* Tab and click on the icon *Create BAdI implementation*.

Enhancement Spot: *RSOA\_VARIABLES\_HANA\_EXIT*

Short Text: *OLAP variables HANA exit*

Composite Enhancement Spot: *ZEAN\_VARIABLE\_HANA\_EXIT*

*Access Key* - 26019554930403899396



*Object can only be created in SAP package*

If you want to edit the object ELEM 00O2SPCKTAN9FU9JJTONQD8BW in this SAP System, have your system administator set the SAP System to "modifiable" for this object.

This can affect the modifiability of the namespace /0CUST/ or the namespaces that correspond to the pattern /0CUST/, as well as the global setting of the system change option.

The system change option is set using the Transport Organizer tools (Transaction SE03). Expand the *Administration* node and execute the program *Set system change option*. The options are described there.

## ***Step 3*** - *Implement the class for the YTD variables*

To get results we’re using a help method \_CALCULATE\_YTD with year as offset and an indicator for from and to values.

To be able to handle shift years we’re using the script function ADD\_YEARS.

In order to be able to process our HANA step-2 variables we have to store the input value of our input variable (CMVM\_CALDAY\_YTD) in method GET\_PROPERTIES in table C\_TS\_VNAM\_INDEX.

class **ZXX\_CL\_VARIABLE\_HANA** *definition*

public final create public .

public section.

interfaces IF\_BADI\_INTERFACE .

interfaces IF\_AMDP\_MARKER\_HDB .

interfaces IF\_RSROA\_VAR\_HANA\_EXIT .

PROTECTED SECTION.

private section.

class-methods: ***\_CALCULATE\_YTD***

importing value(IV\_VALUE1) type RSCHAVL

value(IV\_VALUE\_OFFSET) type INTEGER default 0 *" 0=CY, 1=PY*

value(IV\_VALUE\_TO) type CHAR1 default '' " *''=FR, 'X'=TO*

exporting value(EV\_VALUE) type CHAR8 *"Day .*

ENDCLASS.

CLASS **ZXX\_CL\_VARIABLE\_HANA** I*MPLEMENTATION*.

method ***IF\_RSROA\_VAR\_HANA\_EXIT~GET\_PROPERTIES***

by database procedure for hdb language sqlscript.

C\_IS\_ACTIVE := 'X';

if :I\_VNAM = 'CHVM\_CALDAY\_YTD\_CY\_FR' or

:I\_VNAM = 'CHVM\_CALDAY\_YTD\_CY\_TO' or

:I\_VNAM = 'CHVM\_CALDAY\_YTD\_PY\_FR' or

:I\_VNAM = 'CHVM\_CALDAY\_YTD\_PY\_TO'

then

C\_TS\_VNAM\_INDEX = select 'CMVM\_CALDAY\_YTD' as VNAM,

1 AS INDEX FROM DUMMY;

end if;

endmethod.

method ***IF\_RSROA\_VAR\_HANA\_EXIT~PROCESS***

by database procedure for hdb language sqlscript

using ZXX\_CL\_VARIABLE\_HANA=>\_CALCULATE\_YTD.

if :I\_VNAM = 'CHVM\_CALDAY\_YTD\_CY\_FR'

then call "ZXX\_CL\_VARIABLE\_HANA=>\_CALCULATE\_YTD"( :i\_var\_value\_1, 0 , '' , :c\_value );

end if;

if :I\_VNAM = 'CHVM\_CALDAY\_YTD\_CY\_TO'

then call "ZXX\_CL\_VARIABLE\_HANA=>\_CALCULATE\_YTD"( :i\_var\_value\_1, 0 , 'X' , :c\_value );

endif;

if :I\_VNAM = 'CHVM\_CALDAY\_YTD\_PY\_TO'

then call "ZXX\_CL\_VARIABLE\_HANA=>\_CALCULATE\_YTD"( :i\_var\_value\_1, 1 , 'X' , :c\_value );

end if;

if :I\_VNAM = 'CHVM\_CALDAY\_YTD\_PY\_FR'

then call "ZXX\_CL\_VARIABLE\_HANA=>\_CALCULATE\_YTD"( :i\_var\_value\_1, 1 , '' , :c\_value );

end if;

endmethod.

method ***\_CALCULATE\_YTD***

by database procedure for hdb language sqlscript.

*--Declare Area DECLARE lv\_year INT;*

declare LV\_MON\_FR string;

declare LV\_MON\_TO string;

declare LV\_DAY\_FR string;

declare LV\_DAY\_TO string;

*--Prepare*

LV\_YEAR := substring( ADD\_YEARS( TO\_DATE( :IV\_VALUE1 , 'YYYYMMDD' ) , ( 0 - IV\_VALUE\_OFFSET ) ) , 1, 4) ;

LV\_MON\_TO := substring( ADD\_YEARS( TO\_DATE( :IV\_VALUE1 , 'YYYYMMDD' ) , ( 0 - IV\_VALUE\_OFFSET ) ) , 6, 2) ;

LV\_DAY\_TO := substring( ADD\_YEARS( TO\_DATE( :IV\_VALUE1 , 'YYYYMMDD' ) , ( 0 - IV\_ VALUE\_OFFSET ) ) , 9, 2) ;

LV\_MON\_FR := '01';

LV\_DAY\_FR := '01';

*--Execute*

EV\_VALUE := LV\_YEAR || LV\_MON\_FR || LV\_DAY\_FR ;

if IV\_VALUE\_TO = 'X'

then EV\_VALUE := LV\_YEAR || LV\_MON\_TO || LV\_DAY\_TO ;

end if;

endmethod.

ENDCLASS.

## ***Step 4*** - *Create the SAP HANA exit variables* with BW Query Designer

* CHVM\_CALDAY\_YTD\_CY\_FR – calendar day, year to date current year from
* CHVM\_CALDAY\_YTD\_CY\_TO – calendar day, year to date current year to
* CHVM\_CALDAY\_YTD\_PY\_FR – calendar day, year to date previous year from
* CHVM\_CALDAY\_YTD\_PY\_TO – calendar day, year to date previous year to
* CMVM\_CALDAY\_YTD – calendar day, input ready

Exception *CX\_SY\_NO\_HANDLER* was raised

An exception with the type CX\_AMDP\_VERSION\_MISMATCH was raise

When an exception of type CX\_SY\_NO\_HANDLER is raised, this indicates a programming error within a procedure, where the programmer forgot to prevent an exception of category CX\_DYNAMIC\_CHECK or CX\_STATIC\_CHECK, handle it locally, or declare it.

When exceptions of type CX\_SY\_NO\_HANDLER are handled, it is therefore best not to try to handle the original exception and inform the person responsible for the program instead.

See

# captainabap/[ZCL\_VAR\_HANA\_EXIT.abap](https://gist.github.com/captainabap/71f637ffd347e263b1c081f6fa060047)

# <https://gist.github.com/captainabap/71f637ffd347e263b1c081f6fa060047>

[Содержание](#Содержание)

'Concatenated: ' || :i\_var\_value\_1 || :i\_var\_value\_2;

CLASS **ZCL\_VAR\_HANA\_EXIT** IMPLEMENTATION.

METHOD ***IF\_RSROA\_VAR\_HANA\_EXIT~GET\_PROPERTIES***

*by database procedure for hdb language SQLSCRIPT.*

IF : i\_vnam = *'ZJB\_CONCAT'* THEN

c\_is\_active := 'X';

*-- Table c\_ts\_vnam\_index*

c\_ts\_vnam\_index = select 'ZJB\_VAL1' as *vnam*, 1 as *index* from dummy

union

select 'ZJB\_VAL2' as *vnam*, 2 as *index* from dummy;

END IF;

ENDMETHOD.

METHOD ***IF\_RSROA\_VAR\_HANA\_EXIT~PROCESS***

*by database procedure for hdb language SQLSCRIPT.*

IF :i\_vnam = *'ZJB\_CONCAT'* THEN

*c\_value* := 'Concatenated: ' || :i\_var\_value\_1 || :i\_var\_value\_2;

END IF;

ENDMETHOD.

ENDCLASS.

[Содержание](#Содержание)

from month /07/ to variable /007/

METHOD ***if\_rsroa\_var\_hana\_exit***~***process***

BY DATABASE PROCEDURE FOR HDB

LANGUAGE SQLSCRIPT .

*c\_value := ' '*;

if :i\_vnam = *'ZLP\_CD\_P'* then

if extract ( month from current\_Date ) >= '07' THEN

*c\_valu*e := extract ( year from current\_Date ) + 1;

else

*c\_value* := extract ( year from current\_Date );

end if;

end if;

if :i\_vnam = *'ZHE\_FSPRD'* then

if extract ( month from current\_Date ) = ' 01' then *c\_value* : = '007'; end if;

endif;



METHOD ***GetDatesBetween***

BY DATABASE FUNCTION FOR HDB LANGUAGE SQLSCRIPT.

return

SELECT -- '060' as client,

element\_number as number,

dats\_from\_date( generated\_period\_start ) as date

FROM SERIES\_GENERATE\_DATE('INTERVAL 1 DAY', :starting\_date, add\_days(:end\_date, 1));

ENDMETHOD.



class **ZCL\_PP\_UTILS** *definition*

…

public section.

interfaces IF\_AMDP\_MARKER\_HDB .

class-methods ***getSalesRatio*** for table function ZPP\_SalesRatio.

class-methods ***checkSalesRatio***.

protected section.

private section.

ENDCLASS.

CLASS **ZCL\_PP\_UTILS** *IMPLEMENTATION*.

method ***getSalesRatio***

BY DATABASE FUNCTION FOR HDB LANGUAGE SQLSCRIPT USING /BIC/ABDNPSPD2.

return

select *100* as client,

curr."/BIC/BEMIT000" as EmitterCode,

curr.calday as CurrCalday,

sum(curr."/BIC/BSSLS000") as CurrSum,

prev.calday as PrevCalday,

sum(prev."/BIC/BSSLS000") as PrevSum,

cast(sum(curr."/BIC/BSSLS000") \* 100 / sum(prev."/BIC/BSSLS000") as decimal(18,2)) as Percent

from "/BIC/ABDNPSPD2" as curr

left outer join "/BIC/ABDNPSPD2" as prev

on prev."/BIC/BEMIT000" = curr."/BIC/BEMIT000"

*\*where curr.calday = add\_days(current\_date, -1) and prev.calday = add\_days(current\_date, -8)*

where curr.calday = 20230411 and prev.calday = 20230412

group by curr."/BIC/BEMIT000", curr.calday, prev.calday;

endmethod.

method ***checkSalesRatio***.

SELECT \* FROM *ZPP\_SalesRatio* INTO TABLE @data(it).

WRITE :/ 'Sales ratio'.

loop at it assigning field-symbol(<fs>).

if ( <fs>-percent < 50 ).

WRITE :/ <fs>-emittercode, <fs>-percent.

endif.

endloop.

endmethod.

ENDCLASS.

**My**

class **ZHE\_CL\_EI\_HBEG** *definition*

public

final

create public .

public section.

interfaces IF\_BADI\_INTERFACE.

interfaces IF\_AMDP\_MARKER\_HDB.

interfaces IF\_RSROA\_VAR\_HANA\_EXIT.

protected section.

private section.

ENDCLASS.

CLASS **ZHE\_CL\_EI\_HBEG** *IMPLEMENTATION*.

method ***IF\_RSROA\_VAR\_HANA\_EXIT~GET\_PROPERTIES***

by database procedure for hdb language sqlscript.

C\_IS\_ACTIVE := 'X';

if :i\_vnam = 'ZFISCYEAR\_PCC\_01' then

c\_ts\_vnam\_index = select 'ZFISCYEAR\_PCC\_01' as vnam, 1 as index from dummy

union select 'ZVERSION\_PCU\_01' as vnam, 2 as index from dummy;

end if;

if :I\_VNAM = 'CHVM\_CALDAY\_YTD\_CY\_FR' or

:I\_VNAM = 'CHVM\_CALDAY\_YTD\_CY\_TO' or

:I\_VNAM = 'CHVM\_CALDAY\_YTD\_PY\_FR' or

:I\_VNAM = 'CHVM\_CALDAY\_YTD\_PY\_TO'

then

*-- Input variable CMVM\_CALDAY\_YTD*

*-- Table C\_TS\_VNAM\_INDEX*

C\_TS\_VNAM\_INDEX = select 'CMVM\_CALDAY\_YTD' as VNAM, 1 AS INDEX FROM DUMMY;

end if;

endmethod.

method ***IF\_RSROA\_VAR\_HANA\_EXIT~PROCESS***

by database procedure for hdb language sqlscript.

declare dt date;

DECLARE LV\_LTF\_FFY NVARCHAR(4);

*/\**

*SELECT ZFIRSTFY INTO LV\_LTF\_FFY*

*FROM ZBPCZVERSION WHERE ZVERSION = :i\_var\_value\_2 AND OBJVERS = 'A';*

*c\_value := '';*

*IF :i\_vnam = 'ZFISCYEAR\_PCC\_01' THEN c\_value := LV\_LTF\_FFY;*

*END IF;*

*\*/*

if :I\_VNAM = 'ZAC\_CALDAY\_FROM' then

dt = add\_days(current\_date, -8);

c\_value := to\_dats(dt);

end if;

if :I\_VNAM = 'ZAC\_CALDAY\_TO' then

--dt = add\_days(current\_date, -1);

c\_value:= to\_dats(add\_days(current\_date, -1));

end if;

*--SQL Script for AMDP - Data Declaration & Selects*

*--https://discoveringabap.com/2023/06/17/abap-managed-database-proceduresamdp-sql-script-basics/*

*--Date function in SAP-ABAP-AMDP, calculation of date*

*--https://programmersought.com/article/118311576000/*

endmethod.