[Содержание](#Содержание)

[SimpleTransformation](#SimpleTransformation)

**DATA**: title\_1(10) *VALUE* 'Tutorials'.

**CONCATENATE** '20' DATE0+6(2) DATE0+3(2) DATE0+0(2) *into* data(st).

**CONDENSE** <строка>. - удаляет лишние пробелы между полями, оставляя только 1 пробел.

**CONDENSE** <строка> *NO-GAPS*. - удаляет все пробелы.

**STRLEN**(<строка>).

**SimpleTransformation**

[Содержание](#Содержание)

# Курсы валют на дату

# <http://extsys-bank-by.it.beloil.by/Services/XmlExRates.aspx?ondate=mm/dd/yyyyy>

# 

# se11

# Структура ZNBRB\_RTS

@EndUserText.label : 'ZNBRB\_RTS'

@AbapCatalog.enhancementCategory : #NOT\_CLASSIFIED

define type **znbrb\_rts** {

id : abap.int4;

numcode : abap.char(3);

charcode : abap.char(3);

scale : abap.int4;

name : abap.string(0);

rate : abap.dec(10,4);

# }

# Тип таблицы ZNBRB\_RTS\_T

# 

# Структура ZNBRB

@EndUserText.label : 'ZNBRB'

@AbapCatalog.enhancementCategory : #NOT\_CLASSIFIED

define type **znbrb** {

date : abap.char(10);

currencies : znbrb\_rts\_t;

# }

# Трансформация ZNBRB\_RATES\_3

<?sap.transform simple?>

<tt:transform xmlns:tt="http://www.sap.com/transformation-templates" xmlns:ddic="http://www.sap.com/abapxml/types/dictionary" xmlns:def="http://www.sap.com/abapxml/types/defined">

<tt:root name="DAILYEXRATES" type="ddic:ZNBRB"/>

<tt:template>

<DailyExRates>

*<tt:attribute name="Date*" value-ref=".DAILYEXRATES.DATE"/>

<tt:loop ref=".DAILYEXRATES.CURRENCIES">

<Currency>

*<tt:attribute name="Id"* value-ref="ID"/>

<NumCode tt:value-ref="NUMCODE"/>

<CharCode tt:value-ref="CHARCODE"/>

<Scale tt:value-ref="SCALE"/>

<Name tt:value-ref="NAME"/>

<Rate tt:value-ref="RATE"/>

</Currency>

</tt:loop>

</DailyExRates>

</tt:template>

# </tt:transform>

# abap – программа ZTEST\_RTS

REPORT **ZNBRB\_RTS**.

TABLES: /BIC/ABDCRRTS02.

data:

lt\_xml TYPE solix\_tab,

ls\_rates TYPE znbrb,

ls\_rates\_t TYPE TABLE of znbrb,

lr\_excep1 TYPE REF TO cx\_st\_match\_element,

lr\_excep2 TYPE REF TO cx\_st\_match\_attribute,

lv\_msg TYPE string.

start-of-selection.

perform load\_cbrf\_rates\_from\_site using sy-datum.

form load\_cbrf\_rates\_from\_site using iv\_date type bldat.

data: lr\_client type ref to if\_http\_client,

lr\_response type ref to if\_http\_response,

l\_response type string,

l\_url type string,

lv\_datetext(10) type c,

lv\_url\_add(100) type c.

data: lv\_xml\_string type string,

lv\_date(10) type c.

DATA : lv\_err\_string TYPE string,

lv\_ret\_code TYPE sy-subrc.

data: st(2) type c.

clear lv\_datetext.

if not iv\_date is initial.

write iv\_date to lv\_datetext.

lv\_datetext+2(1) = lv\_datetext+5(1) = '/'.

st = lv\_datetext+0(2). " dd

lv\_datetext+0(2) = lv\_datetext+3(2). " mm/mm/yyyy

lv\_datetext+3(2) = st. " dd/mm/yyyy

\* write: / lv\_datetext.

concatenate '?ondate=' lv\_datetext into lv\_url\_add.

endif.

concatenate 'http://extsys-bank-by.it.beloil.by/Services/XmlExRates.aspx' lv\_url\_add into l\_url.

\*write: / l\_url.

call method cl\_http\_client=>create\_by\_url

exporting url = l\_url

importing client = lr\_client

exceptions

internal\_error = 1

argument\_not\_found = 2

plugin\_not\_active = 3

others = 4.

if sy-subrc ne 0.

write: / 'Ошибка обращения к серверу /cl\_http\_client=>create\_by\_url/'.

" message e002 raising create\_url\_err.

endif.

\* Отправка запроса

call method lr\_client->send

\* exporting timeout = timeout

exceptions

http\_communication\_failure = 1

http\_invalid\_state = 2

http\_processing\_failed = 3

others = 4.

if sy-subrc ne 0.

write: / 'Ошибка запроса к серверу /lr\_client->send/'.

" message e003 raising send\_error.

endif.

\* Получение ответа сервера

call method lr\_client->receive

exceptions

http\_communication\_failure = 1

http\_invalid\_state = 2

http\_processing\_failed = 3

others = 4.

if sy-subrc ne 0.

lr\_client->response->get\_status(

IMPORTING

code = lv\_ret\_code

reason = lv\_err\_string

).

\* MESSAGE lv\_err\_string TYPE 'I'.

write: / 'Ошибка получения ответа от сервера /lr\_client->receive/'.

"message e004 raising receive\_error.

endif.

lr\_response = lr\_client->response .

l\_response = lr\_response->get\_cdata( ).

\* Закрытие соединения

call method lr\_client->close

exceptions

http\_invalid\_state = 1

others = 2.

check not l\_response is initial.

lv\_xml\_string = l\_response.

\*write : / lv\_xml\_string.

TRY.

CALL TRANSFORMATION znbrb\_rates\_3

SOURCE XML lv\_xml\_string

RESULT DAILYEXRATES = ls\_rates.

CATCH cx\_st\_match\_element INTO lr\_excep1.

lv\_msg = lr\_excep1->get\_text( ).

CATCH cx\_st\_match\_attribute INTO lr\_excep2.

lv\_msg = lr\_excep2->get\_text( ).

ENDTRY.

FIELD-SYMBOLS: <fs> TYPE znbrb\_rts.

\*write: / ls\_rates-date.

concatenate ls\_rates-date+6(4) ls\_rates-date+0(2) ls\_rates-date+3(2) into data(dats).

write: / dats.

types: begin of struct\_01,

data0 type c length 8,

f\_n\_code type n length 3,

f\_c\_code type c length 3,

f\_dscr type c length 100,

f\_scale type int4,

f\_ratio type f,

end of struct\_01.

data:

intab\_01 type table of struct\_01,

wa like line of intab\_01.

data:

l\_lines\_inserted type int4,

lt\_msg type rs\_t\_msg,

datum type dats.

datum = dats.

CALL FUNCTION 'OIU\_ME\_CHAR\_TO\_DATE' "E&P measurement: convert character to date

EXPORTING

i\_char = dats " c

IMPORTING

e\_date = datum " d

EXCEPTIONS

ERROR\_FOUND = 1.

if sy-subrc = 0.

write: / 'datum - ', datum.

endif.

LOOP AT ls\_rates-currencies ASSIGNING <fs>.

IF <fs>-numcode = 840.

\* write: / ' | NumCode - ', <fs>-numcode, ' | CharCode - ', <fs>-charcode, ' ', <fs>-name, '| Scale - ', <fs>-scale, ' | Rate - ', <fs>-rate.

ENDIF.

\* wa-data0 = dats.

wa-data0 = datum.

wa-f\_n\_code = <fs>-numcode. wa-f\_c\_code = <fs>-charcode. wa-f\_dscr = <fs>-name. wa-f\_scale = <fs>-scale. wa-f\_ratio = <fs>-rate.

insert wa into table intab\_01.

ENDLOOP.

loop at intab\_01 into wa.

write: / wa-f\_scale, ' ', wa-f\_dscr+0(30), ' ', wa-f\_ratio.

endloop.

CALL FUNCTION 'RSDSO\_WRITE\_API'

EXPORTING

i\_adsonm = 'BDCRRTS0'

i\_allow\_new\_sids = rs\_c\_false

i\_activate\_data = rs\_c\_true

it\_data = intab\_01

IMPORTING

e\_lines\_inserted = l\_lines\_inserted

et\_msg = lt\_msg

\* e\_upd\_req\_tsn = l\_upd\_req

\* et\_act\_req\_tsn = lt\_act\_req

EXCEPTIONS

write\_failed = 1

activation\_failed = 2

datastore\_not\_found = 3

OTHERS = 4.

skip.

case sy-subrc.

when 1. write: / 'RSDSO\_WRITE\_API return - write\_failed'.

when 2. write: / 'RSDSO\_WRITE\_API return -activation\_failed'.

when 3. write: / 'RSDSO\_WRITE\_API return - datastore\_not\_found'.

endcase.

skip. write: / l\_lines\_inserted, ' lines are was inserted'.

# endform. "load\_cbrf\_rates\_from\_site

# [Sample Code for Expert Routine](https://wiki.scn.sap.com/wiki/display/BI/Sample+Code+for+Expert+Routine)

<https://wiki.scn.sap.com/wiki/display/BI/Sample+Code+for+Expert+Routine>

METHOD expert\_routine.  
*\*=== Segments ===*  
  
 FIELD-SYMBOLS:  
 <SOURCE\_FIELDS> TYPE \_ty\_s\_SC\_1.  
  
 DATA:  
 RESULT\_FIELDS TYPE \_ty\_s\_TG\_1.  
  
*\*$\*$ begin of routine - insert your code only below this line \*-\**  
... *"insert your code here*  
DATA: DATE0 TYPE D,  
 F\_CH(100) TYPE c,  
 BNP\_TYPE(4) TYPE n,  
 F\_CH\_VAL(10) TYPE c.  
  
LOOP at SOURCE\_PACKAGE into <SOURCE\_FIELDS>.  
 DATE0 = <SOURCE\_FIELDS>-DATE\_.  
 F\_CH = <SOURCE\_FIELDS>-OP\_CH.  
 BNP\_TYPE = '0001'.  
 F\_CH\_VAL = <SOURCE\_FIELDS>-F\_80.  
  
 CONCATENATE '20' DATE0+6(2) DATE0+3(2) DATE0+0(2) into DATE0.  
 RESULT\_FIELDS-DATE0 = DATE0.  
  
 REPLACE '%' WITH '' INTO F\_CH.  
 CONDENSE F\_CH NO-GAPS.  
 RESULT\_FIELDS-F\_CH = F\_CH.  
 RESULT\_FIELDS-/BIC/BNP\_TYPE = BNP\_TYPE.  
 RESULT\_FIELDS-F\_CH\_VAL = F\_CH\_VAL.  
  
 append RESULT\_FIELDS to RESULT\_PACKAGE.  
  
 clear DATE0.  
 clear F\_CH.  
 clear BNP\_TYPE.  
 clear F\_CH\_VAL.  
ENDLOOP.  
*\*$\*$ end of routine - insert your code only before this line \*-\**  
 ENDMETHOD.

# Converting ABAP code into AMDP scripts in SAP BW transformations

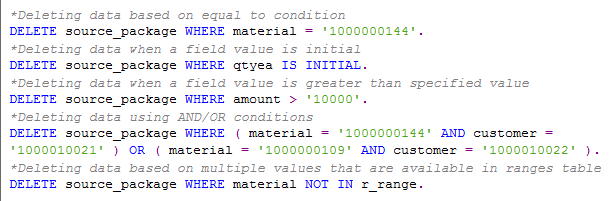
<https://visualbi.com/blogs/sap/sap-bw-hana/converting-abap-code-amdp-scripts-sap-bw-transformations/>

Unlike ABAP transformation, where row level processing is possible, in AMDP, data is processed as a package using an INSERT AS SELECT on the calculation scenario generated.

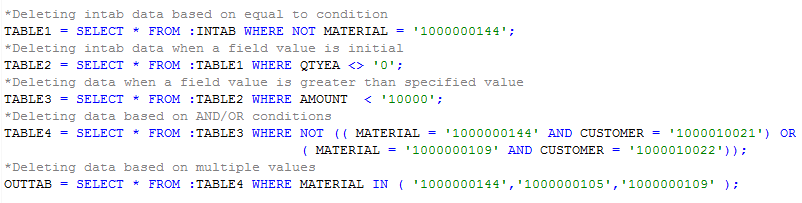
### **Delete**

In ABAP, the **DELETE** statement is used to remove certain records from an internal table based on certain conditions. In the AMDP script, the DELETE statement is replaced by SELECT statements.

***ABAP***



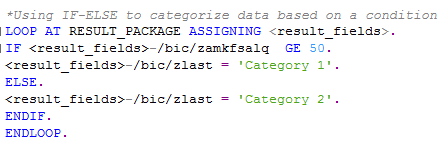
***AMDP***



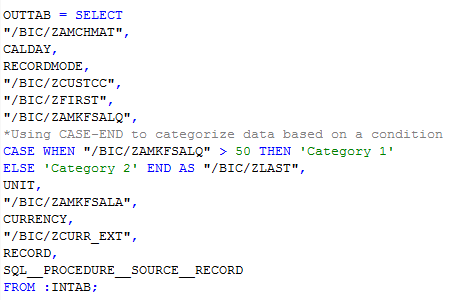
### **IF-ELSE Statements**

The IF-ELSE-ENDIF statements are used in ABAP to process different logical blocks of code based on different conditions. In AMDP code, the IF-ELSE statements can be replaced by CASE-END statements.

***ABAP***



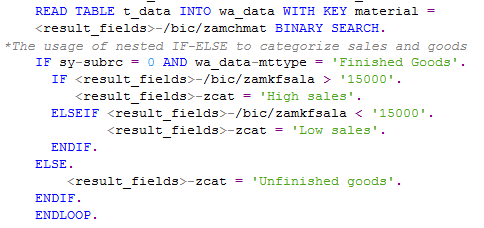
***AMDP***



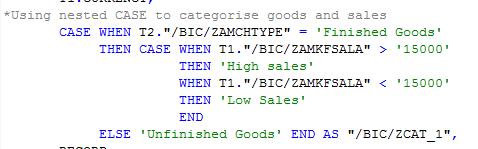
Example 2

Nested if-else can also be commonly seen as a part of ABAP code and CASE statements can be used to replace nested if-else statements as well. An example with nested CASE is shown below.

***ABAP***



***AMDP***

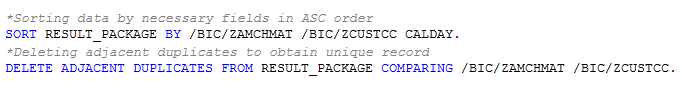


### **Sort, Delete adjacent duplicates and For all entries**

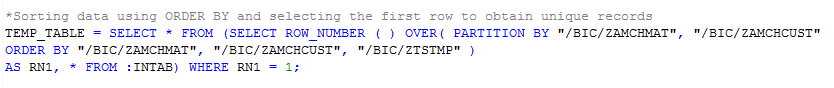
Sort statements can be replaced by the “ORDER BY” clause in AMDP.

Delete adjacent duplicates in ABAP is achieved in the AMDP script using the ROW\_NUMBER() and OVER() functions. The “PARTITION BY” clause divides the data into groups and ROW\_NUMBER() clause assigns a row number for the partitioned data. If the partition by clause is not used, then the entire set is considered as a single group of data.

***ABAP***



***AMDP***



### **For all entries**

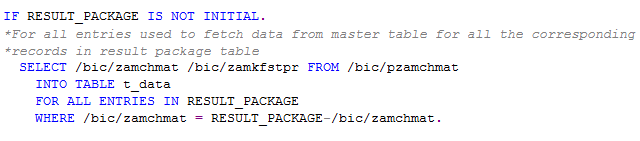
In ABAP, FOR ALL ENTRIES statement is used to fetch corresponding data for all the records in the source or result package. The same functionality can be achieved using “LEFT OUTER JOIN” in the AMDP script.

*Rem*

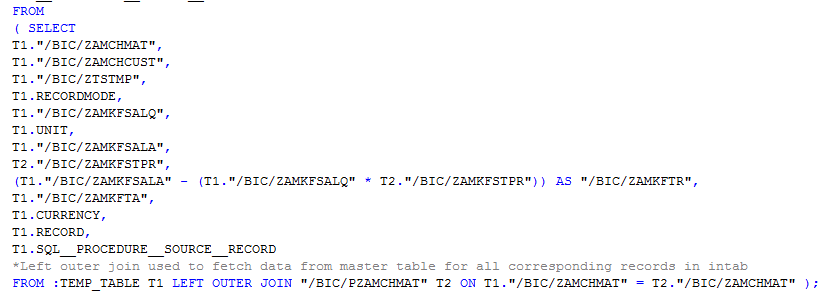
An important point to be noted during joins in AMDP – If any table other than intab/outtab is to be used in AMDP join, then the table name must be specified in the ‘METHOD PROCEDURE’ statement along with the “USING” clause.

Converting ABAP code into AMDP scripts in SAP BW transformations

***ABAP***



***AMDP***



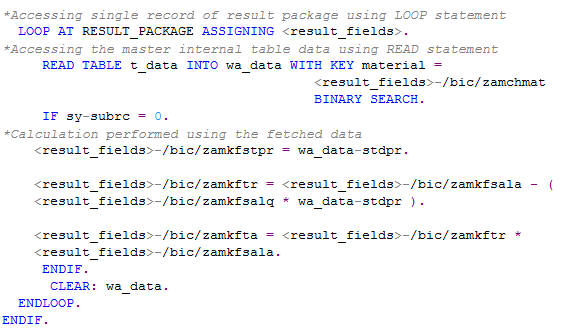
### **Loop and Read statements**

In ABAP, single record processing of data in the source and result packages is achieved with the help of LOOP-ENDLOOP and READ TABLE statements. Since LOOP-ENDLOOP along with READ statements are not compatible with AMDP script, subqueries are used to replace them.

*Note*

A calculated field cannot be used for the calculation of another field within the same SELECT. The calculation is to be carried out in the subquery and then the calculated field may be used for another calculation in the main select.

***ABAP***



***AMDP***

