****

**ABAP Development for SAP HANA**

[**https://open.sap.com/courses/a4h1**](https://open.sap.com/courses/a4h1)

**Source Code of Week 4**



Version 1 - 15.10.2014

Only Demo content

Without any warranty

Contents

[CDS DDL Sources 3](#_Toc401149124)

[ZDDLS\_CDS\_40\_ANNOTATION 3](#_Toc401149125)

[ABAP Classes 3](#_Toc401149126)

[ZCL\_AMDP\_SIMPLE\_00 3](#_Toc401149127)

[ZCL\_A4H1\_AMDP\_BADI 4](#_Toc401149128)

[ZCL\_A4H1\_BADI\_IMPL\_01 5](#_Toc401149129)

[ZCL\_A4H1\_BADI\_IMPL\_02 6](#_Toc401149130)

[ZCL\_A4H1\_CALL\_AMDP\_BADI 7](#_Toc401149131)

[ZCL\_A4H1\_E2E\_CUST\_OPEN\_INVOICE 8](#_Toc401149132)

[ZCL\_ZA4H1\_E2E\_CUST\_INF\_DPC\_EXT 12](#_Toc401149133)

[ABAP Programs 15](#_Toc401149134)

[ZR\_ADBC\_PROC\_CALL 15](#_Toc401149135)

[ZR\_ADBC\_SIMPLE 17](#_Toc401149136)

[ZR\_AMDP\_01\_SIMPLE\_CALL 18](#_Toc401149137)

[ZR\_AMDP\_BADI\_CALL 19](#_Toc401149138)

[ZR\_AMDP\_CALL 19](#_Toc401149139)

[ABAP Interfaces 19](#_Toc401149140)

[ZIF\_A4H1\_AMDP\_BADI 19](#_Toc401149141)

# CDS DDL Sources

## ZDDLS\_CDS\_40\_ANNOTATION

@AbapCatalog.sqlViewName: 'ZDDLS\_CDS\_40'

@ClientDependent: true

@AbapCatalog.Buffering.status: #SWITCHED\_OFF

**define** **view** zcdsv\_annotation\_simple **as** **select** **from** snwd\_so

**{**

**key** so\_id **as** customer\_id**,**

@Semantics.currencyCode: true

currency\_code**,**

@Semantics.amount.currencyCode: 'currency\_code'

gross\_amount

**}**

# ABAP Classes

## ZCL\_AMDP\_SIMPLE\_00

CLASS zcl\_amdp\_simple\_00 DEFINITION

PUBLIC

FINAL

CREATE PUBLIC .

PUBLIC SECTION.

INTERFACES: if\_amdp\_marker\_hdb.

TYPES:

BEGIN OF ty\_customer\_info,

customer\_id TYPE snwd\_bpa-bp\_id,

customer\_name TYPE snwd\_bpa-company\_name,

currency\_code TYPE snwd\_so-currency\_code,

total\_gross\_amount TYPE snwd\_so-gross\_amount,

undefined TYPE i,

END OF ty\_customer\_info ,

tt\_customer\_info TYPE STANDARD TABLE OF ty\_customer\_info.

METHODS get\_customer\_infos

EXPORTING

VALUE(et\_customer\_info) TYPE tt\_customer\_info

RAISING cx\_amdp\_error.

ENDCLASS.

CLASS ZCL\_AMDP\_SIMPLE\_00 IMPLEMENTATION.

METHOD get\_customer\_infos

by DATABASE PROCEDURE

FOR HDB

LANGUAGE SQLSCRIPT

options read-ONLY

using snwd\_bpa snwd\_so.

declare lv\_zero integer;

lv\_zero := 0;

et\_customer\_info =

SELECT BP\_ID as customer\_id,

COMPANY\_NAME as customer\_name,

SO.CURRENCY\_CODE,

SUM( SO.GROSS\_AMOUNT )

as TOTAL\_GROSS\_AMOUNT

,

( 1 / :lv\_zero ) as undefined

FROM SNWD\_BPA AS BPA

INNER JOIN SNWD\_SO AS SO

ON SO.BUYER\_GUID = BPA.NODE\_KEY

GROUP BY BP\_ID, COMPANY\_NAME, SO.CURRENCY\_CODE

ORDER BY BP\_ID;

ENDMETHOD.

ENDCLASS.

## ZCL\_A4H1\_AMDP\_BADI

CLASS zcl\_a4h1\_amdp\_badi DEFINITION

PUBLIC

FINAL

CREATE PUBLIC .

PUBLIC SECTION.

INTERFACES if\_amdp\_marker\_hdb.

INTERFACES if\_badi\_interface .

INTERFACES zif\_a4h1\_amdp\_badi .

PROTECTED SECTION.

PRIVATE SECTION.

ENDCLASS.

CLASS ZCL\_A4H1\_AMDP\_BADI IMPLEMENTATION.

METHOD zif\_a4h1\_amdp\_badi~get\_customer\_infos

BY DATABASE PROCEDURE

FOR HDB

LANGUAGE SQLSCRIPT

OPTIONS READ-ONLY

USING snwd\_so snwd\_bpa.

et\_customer\_info =

SELECT BP\_ID as customer\_id,

COMPANY\_NAME as customer\_name,

SO.CURRENCY\_CODE,

SUM( SO.GROSS\_AMOUNT )

as TOTAL\_GROSS\_AMOUNT

FROM SNWD\_BPA AS BPA

INNER JOIN SNWD\_SO AS SO

ON SO.BUYER\_GUID = BPA.NODE\_KEY

GROUP BY BP\_ID, COMPANY\_NAME, SO.CURRENCY\_CODE

ORDER BY BP\_ID;

ENDMETHOD.

ENDCLASS.

## ZCL\_A4H1\_BADI\_IMPL\_01

CLASS zcl\_a4h1\_badi\_impl\_01 DEFINITION

PUBLIC

FINAL

CREATE PUBLIC .

PUBLIC SECTION.

INTERFACES if\_amdp\_marker\_hdb.

INTERFACES if\_badi\_interface .

INTERFACES zif\_a4h1\_amdp\_badi .

PROTECTED SECTION.

PRIVATE SECTION.

ENDCLASS.

CLASS zcl\_a4h1\_badi\_impl\_01 IMPLEMENTATION.

METHOD zif\_a4h1\_amdp\_badi~get\_customer\_infos

BY DATABASE PROCEDURE

FOR HDB

LANGUAGE SQLSCRIPT

OPTIONS READ-ONLY

USING snwd\_so snwd\_bpa.

et\_customer\_info =

SELECT BP\_ID as customer\_id,

COMPANY\_NAME as customer\_name,

SO.CURRENCY\_CODE,

SUM( SO.GROSS\_AMOUNT )

as TOTAL\_GROSS\_AMOUNT

FROM SNWD\_BPA AS BPA

INNER JOIN SNWD\_SO AS SO

ON SO.BUYER\_GUID = BPA.NODE\_KEY

GROUP BY BP\_ID, COMPANY\_NAME, SO.CURRENCY\_CODE

ORDER BY BP\_ID;

ENDMETHOD.

ENDCLASS.

## ZCL\_A4H1\_BADI\_IMPL\_02

CLASS zcl\_a4h1\_badi\_impl\_02 DEFINITION

PUBLIC

FINAL

CREATE PUBLIC .

PUBLIC SECTION.

INTERFACES if\_amdp\_marker\_hdb.

INTERFACES if\_badi\_interface .

INTERFACES zif\_a4h1\_amdp\_badi .

PROTECTED SECTION.

PRIVATE SECTION.

ENDCLASS.

CLASS zcl\_a4h1\_badi\_impl\_02 IMPLEMENTATION.

METHOD zif\_a4h1\_amdp\_badi~get\_customer\_infos

BY DATABASE PROCEDURE

FOR HDB

LANGUAGE SQLSCRIPT

OPTIONS READ-ONLY

USING snwd\_so snwd\_bpa.

et\_customer\_info =

SELECT BP\_ID as customer\_id,

COMPANY\_NAME as customer\_name,

SO.CURRENCY\_CODE,

SUM( SO.GROSS\_AMOUNT )

as TOTAL\_GROSS\_AMOUNT

FROM SNWD\_BPA AS BPA

INNER JOIN SNWD\_SO AS SO

ON SO.BUYER\_GUID = BPA.NODE\_KEY

GROUP BY BP\_ID, COMPANY\_NAME, SO.CURRENCY\_CODE

HAVING SUM( SO.GROSS\_AMOUNT ) > 10000000

ORDER BY BP\_ID;

ENDMETHOD.

ENDCLASS.

## ZCL\_A4H1\_CALL\_AMDP\_BADI

CLASS zcl\_a4h1\_call\_amdp\_badi DEFINITION

PUBLIC

FINAL

CREATE PUBLIC .

PUBLIC SECTION.

INTERFACES: if\_amdp\_marker\_hdb.

class-METHODS:

call\_badi

EXPORTING value(et\_customer\_info) type zif\_a4h1\_amdp\_badi=>tt\_customer\_info.

PROTECTED SECTION.

PRIVATE SECTION.

ENDCLASS.

CLASS ZCL\_A4H1\_CALL\_AMDP\_BADI IMPLEMENTATION.

METHOD call\_badi

by DATABASE PROCEDURE

FOR HDB

LANGUAGE SQLSCRIPT

options read-only

using ZA4H1\_AMDP\_BADI=>get\_customer\_infos.

call "ZA4H1\_AMDP\_BADI=>GET\_CUSTOMER\_INFOS"( et\_customer\_info => et\_customer\_info );

ENDMETHOD.

ENDCLASS.

## ZCL\_A4H1\_E2E\_CUST\_OPEN\_INVOICE

CLASS zcl\_a4h1\_e2e\_cust\_open\_invoice DEFINITION

PUBLIC

FINAL

CREATE PUBLIC .

PUBLIC SECTION.

INTERFACES: if\_amdp\_marker\_hdb.

TYPES:

tt\_cust\_info TYPE STANDARD TABLE OF za4h1\_e2e\_customer\_info WITH KEY customer\_id,

tt\_inv\_info TYPE STANDARD TABLE OF za4h1\_e2e\_invoice\_info WITH KEY customer\_id.

METHODS:

get\_customer\_info

IMPORTING

VALUE(iv\_client) TYPE symandt

VALUE(iv\_bupaid)

TYPE za4h1\_e2e\_customer\_info-customer\_id

EXPORTING

VALUE(et\_bpinfo) TYPE tt\_cust\_info,

get\_invoice\_info

IMPORTING

VALUE(iv\_client) TYPE symandt

VALUE(iv\_bupaid)

TYPE za4h1\_e2e\_customer\_info-customer\_id

EXPORTING

VALUE(et\_invinfo) TYPE tt\_inv\_info.

PROTECTED SECTION.

PRIVATE SECTION.

TYPES:

BEGIN OF ty\_rel\_items,

client TYPE snwd\_so\_inv\_item-client,

inv\_i\_guid TYPE snwd\_so\_inv\_item-node\_key,

inv\_guid TYPE snwd\_so\_inv\_head-node\_key,

buyer\_guid TYPE snwd\_bpa-node\_key,

customer\_id TYPE snwd\_bpa-bp\_id,

invoice\_date TYPE snwd\_so\_inv\_head-created\_at,

gross\_amount

TYPE snwd\_so\_inv\_item-gross\_amount,

currency\_code\_conv

TYPE snwd\_so\_inv\_item-currency\_code,

END OF ty\_rel\_items,

tt\_rel\_items TYPE

STANDARD TABLE OF ty\_rel\_items.

METHODS:

get\_curr\_conv\_relevant\_items

IMPORTING

VALUE(iv\_client) TYPE symandt

VALUE(iv\_bupaid)

TYPE za4h1\_e2e\_customer\_info-customer\_id

EXPORTING

VALUE(et\_conv\_items) TYPE tt\_rel\_items.

ENDCLASS.

CLASS zcl\_a4h1\_e2e\_cust\_open\_invoice IMPLEMENTATION.

METHOD get\_curr\_conv\_relevant\_items BY DATABASE PROCEDURE

FOR HDB

LANGUAGE SQLSCRIPT

OPTIONS READ-ONLY

USING snwd\_bpa snwd\_so\_inv\_head snwd\_so\_inv\_item.

-- declare a local variable

declare lv\_today date;

-- get current date for conversion

select current\_date into lv\_today from dummy;

-- select relevant invoice items

lt\_relevant\_items =

select

i.client as client,

i.node\_key as inv\_i\_guid,

h.node\_key as inv\_guid,

bpa.node\_key as buyer\_guid,

bpa.bp\_id as customer\_id,

h.created\_at as invoice\_date,

i.gross\_amount,

i.currency\_code

from snwd\_so\_inv\_item as i

join snwd\_so\_inv\_head as h

on i.client = h.client

and i.parent\_key = h.node\_key

join snwd\_bpa as bpa

on h.client = bpa.client

and h.buyer\_guid = bpa.node\_key

where h.client = :iv\_client

and bpa.bp\_id = :iv\_bupaid

and h.payment\_status = '';

--convert gross amount of items to currency 'USD'

et\_conv\_items = ce\_conversion( :lt\_relevant\_items,

[ family = 'currency',

method = 'ERP',

steps = 'shift,convert,shift\_back',

source\_unit\_column = "CURRENCY\_CODE" ,

output\_unit\_column = "CURRENCY\_CODE\_CONV",

target\_unit = 'USD',

reference\_date = :lv\_today,

client = :iv\_client ],

[gross\_amount] ) ;

ENDMETHOD.

METHOD get\_invoice\_info BY DATABASE PROCEDURE

FOR HDB

LANGUAGE SQLSCRIPT

OPTIONS READ-ONLY

USING snwd\_so snwd\_so\_inv\_head zcl\_a4h1\_e2e\_cust\_open\_invoice=>get\_curr\_conv\_relevant\_items.

call "ZCL\_A4H1\_E2E\_CUST\_OPEN\_INVOICE=>GET\_CURR\_CONV\_RELEVANT\_ITEMS" (

iv\_client => :iv\_client,

iv\_bupaid => :iv\_bupaid,

et\_conv\_items => :lt\_converted\_items );

--aggregated gross amounts per sales order invoice

et\_invinfo =

select

customer\_id,

so\_id as order\_id,

invoice\_date,

currency\_code\_conv as currency\_code,

sum( conv\_items.gross\_amount ) as sum\_gross\_amount

from :lt\_converted\_items as conv\_items

join snwd\_so\_inv\_head as h

on h.client = conv\_items.client

and h.node\_key = conv\_items.inv\_guid

join snwd\_so as so

on so.client = h.client

and so.node\_key = h.so\_guid

group by customer\_id, so\_id, invoice\_date, currency\_code\_conv

order by so\_id asc;

ENDMETHOD.

METHOD get\_customer\_info BY DATABASE PROCEDURE

FOR HDB

LANGUAGE SQLSCRIPT

OPTIONS READ-ONLY

USING snwd\_bpa snwd\_ad zcl\_a4h1\_e2e\_cust\_open\_invoice=>get\_curr\_conv\_relevant\_items.

call "ZCL\_A4H1\_E2E\_CUST\_OPEN\_INVOICE=>GET\_CURR\_CONV\_RELEVANT\_ITEMS" (

iv\_client => :iv\_client,

iv\_bupaid => :iv\_bupaid,

et\_conv\_items => :lt\_converted\_items );

--aggregated gross amounts per customer

et\_bpinfo =

select

customer\_id,

bpa.company\_name as customer\_name,

ad.city,

ad.street,

ad.postal\_code,

ad.country,

conv\_items.currency\_code\_conv as currency\_code,

sum( conv\_items.gross\_amount ) as sum\_gross\_amount

from :lt\_converted\_items as conv\_items

join snwd\_bpa as bpa

on conv\_items.client = bpa.client

and buyer\_guid = bpa.node\_key

join snwd\_ad as ad

on bpa.client = ad.client

and bpa.address\_guid = ad.node\_key

group by customer\_id, company\_name, city, street, postal\_code, country, currency\_code\_conv;

ENDMETHOD.

ENDCLASS.

## ZCL\_ZA4H1\_E2E\_CUST\_INF\_DPC\_EXT

CLASS zcl\_za4h1\_e2e\_cust\_inf\_dpc\_ext DEFINITION

PUBLIC

INHERITING FROM zcl\_za4h1\_e2e\_cust\_inf\_dpc

CREATE PUBLIC .

PUBLIC SECTION.

PROTECTED SECTION.

METHODS invoiceinfos\_get\_entityset REDEFINITION.

METHODS customerinfos\_get\_entity REDEFINITION.

PRIVATE SECTION.

ENDCLASS.

CLASS zcl\_za4h1\_e2e\_cust\_inf\_dpc\_ext IMPLEMENTATION.

METHOD invoiceinfos\_get\_entityset.

DATA ls\_cust\_classification TYPE zcl\_za4h1\_e2e\_cust\_inf\_mpc=>ts\_customerclassification.

"Entity can only be accessed via the navigation property ToInvoiceInfo

"of entity type CustomerClassification and if the key CustomerId is provided

IF iv\_source\_name <> zcl\_za4h1\_e2e\_cust\_inf\_mpc=>gc\_customerclassification.

RAISE EXCEPTION TYPE /iwbep/cx\_mgw\_tech\_exception.

ENDIF.

io\_tech\_request\_context->get\_converted\_source\_keys(

IMPORTING

es\_key\_values = ls\_cust\_classification ).

"create an instance of class zcl\_customer\_open\_invoices

DATA(lo\_cust\_info) = NEW zcl\_a4h1\_e2e\_cust\_open\_invoice( ).

"call the ABAP managed DB procedure respectively the

"ABAP class method get\_invoice\_info

lo\_cust\_info->get\_invoice\_info(

EXPORTING

iv\_client = sy-mandt

iv\_bupaid = ls\_cust\_classification-customer\_id

IMPORTING

et\_invinfo = DATA(lt\_entityset)

).

"provide the resultset of the AMDP to the tabular

"tabular output parameter

et\_entityset = lt\_entityset.

ENDMETHOD.

METHOD customerinfos\_get\_entity.

DATA ls\_cust\_classification TYPE zcl\_za4h1\_e2e\_cust\_inf\_mpc=>ts\_customerclassification.

"Entity can only be accessed via the navigation property ToCustomerInfo

"of entity type CustomerClassification and if the key CustomerId is provided

IF iv\_source\_name <> zcl\_za4h1\_e2e\_cust\_inf\_mpc=>gc\_customerclassification.

RAISE EXCEPTION TYPE /iwbep/cx\_mgw\_tech\_exception.

ENDIF.

io\_tech\_request\_context->get\_converted\_source\_keys(

IMPORTING

es\_key\_values = ls\_cust\_classification ).

"create an instance of class zcl\_customer\_open\_invoices

DATA(lo\_cust\_info) = NEW zcl\_a4h1\_e2e\_cust\_open\_invoice( ).

"call the ABAP managed DB procedure respectively the

"ABAP class method get\_customer\_info

lo\_cust\_info->get\_customer\_info(

EXPORTING

iv\_client = sy-mandt

iv\_bupaid = ls\_cust\_classification-customer\_id

IMPORTING

et\_bpinfo = DATA(lt\_entity)

).

"only one line is expected to be retrieved

IF lines( lt\_entity ) <> 1.

RAISE EXCEPTION TYPE /iwbep/cx\_mgw\_busi\_exception

EXPORTING

message\_unlimited = | Problem in Customer Info; { lines( lt\_entity ) } <> 1 |

textid = /iwbep/cx\_mgw\_busi\_exception=>business\_error\_unlimited.

ENDIF.

"map the first row of lt\_entity to the output

er\_entity = lt\_entity[ 1 ].

ENDMETHOD.

ENDCLASS.

# ABAP Programs

## ZR\_ADBC\_PROC\_CALL

REPORT zr\_adbc\_proc\_call.

"lot of type definitions

TYPES:

BEGIN OF ty\_overview,

value TYPE string,

table TYPE string,

END OF ty\_overview,

BEGIN OF ty\_invoice\_header,

invoice\_guid TYPE snwd\_so\_inv\_head-node\_key,

created\_at TYPE snwd\_so\_inv\_head-created\_at,

paid\_at TYPE snwd\_so\_inv\_head-changed\_at,

buyer\_guid TYPE snwd\_so\_inv\_head-buyer\_guid,

END OF ty\_invoice\_header .

TYPES:

BEGIN OF ty\_invoice\_item,

item\_guid TYPE snwd\_so\_inv\_item-node\_key,

invoice\_guid TYPE snwd\_so\_inv\_head-node\_key,

product\_guid TYPE snwd\_so\_inv\_item-product\_guid,

gross\_amount TYPE snwd\_so\_inv\_item-gross\_amount,

currency\_code TYPE snwd\_so\_inv\_item-currency\_code,

END OF ty\_invoice\_item .

TYPES:

BEGIN OF ty\_customer\_info,

customer\_guid TYPE snwd\_bpa-node\_key,

customer\_id TYPE snwd\_bpa-bp\_id,

customer\_name TYPE snwd\_bpa-company\_name,

country TYPE snwd\_ad-country,

postal\_code TYPE snwd\_ad-postal\_code,

city TYPE snwd\_ad-city,

END OF ty\_customer\_info .

TYPES:

tt\_invoice\_header TYPE STANDARD TABLE OF ty\_invoice\_header WITH KEY invoice\_guid .

TYPES:

tt\_invoice\_item TYPE STANDARD TABLE OF ty\_invoice\_item .

TYPES:

tt\_customer\_info TYPE STANDARD TABLE OF ty\_customer\_info .

DATA lv\_stmt TYPE string.

DATA lo\_stmt TYPE REF TO cl\_sql\_statement.

DATA lo\_res TYPE REF TO cl\_sql\_result\_set.

DATA lt\_overview TYPE STANDARD TABLE OF ty\_overview.

DATA lt\_inv\_head TYPE tt\_invoice\_header.

DATA lt\_inv\_item TYPE tt\_invoice\_item.

DATA lt\_cust\_info TYPE tt\_customer\_info.

lv\_stmt = | CALL "SAPHANAABAP"."ZCL\_DEMO\_PAID\_ON\_DATE\_AMDP=>PAID\_ON\_DATE" | &&

| ( '20140912', NULL, NULL, NULL ) WITH OVERVIEW |.

TRY.

lo\_stmt = NEW cl\_sql\_statement( ).

lo\_res = lo\_stmt->execute\_query( lv\_stmt ).

"get a reference of the overview table and prepare the result set

lo\_res->set\_param\_table( REF #( lt\_overview ) ).

"retrieve the overview table

lo\_res->next\_package( ).

LOOP AT lt\_overview INTO DATA(ls\_overview).

"select from the corresponding DB table listed in the overview table

DATA(lo\_res\_tab) = lo\_stmt->execute\_query( | select \* from { ls\_overview-table }| ).

IF ls\_overview-value CS 'ET\_INVOICE\_ITEM'.

"prepare the result set

lo\_res\_tab->set\_param\_table( REF #( lt\_inv\_item ) ).

ELSEIF ls\_overview-value CS 'ET\_INVOICE\_HEAD'.

"prepare the result set

lo\_res\_tab->set\_param\_table( REF #( lt\_inv\_head ) ).

ELSEIF ls\_overview-value CS 'ET\_CUSTOMER\_INFO'.

"prepare the result set

lo\_res\_tab->set\_param\_table( REF #( lt\_cust\_info ) ).

ENDIF.

"fetch the content of the database tables into the internal tables

lo\_res\_tab->next\_package( ).

ENDLOOP.

lo\_res->close( ).

CATCH cx\_sql\_exception INTO DATA(lx).

"do some meaningful error handling

WRITE: lx->sql\_message.

ENDTRY.

cl\_demo\_output=>display\_data( name = 'Customer Info' value = lt\_cust\_info ).

## ZR\_ADBC\_SIMPLE

REPORT zr\_adbc\_simple.

TYPES:

BEGIN OF ty\_res,

bp\_id TYPE snwd\_bpa-bp\_id,

company\_name TYPE snwd\_bpa-company\_name,

currency\_code TYPE snwd\_so-currency\_code,

total\_gross\_amount TYPE snwd\_so-gross\_amount,

END OF ty\_res.

DATA lv\_stmt TYPE string.

DATA lo\_stmt TYPE REF TO cl\_sql\_statement.

DATA lo\_res TYPE REF TO cl\_sql\_result\_set.

DATA lt\_result TYPE STANDARD TABLE OF ty\_res WITH EMPTY KEY.

lv\_stmt = |SELECT BP\_ID, COMPANY\_NAME, SO.CURRENCY\_CODE, | &&

| SUM( SO.GROSS\_AMOUNT ) as TOTAL\_GROSS\_AMOUNT | &&

| FROM SNWD\_BPA AS BPA | &&

| INNER JOIN SNWD\_SO AS SO | &&

| ON SO.BUYER\_GUID = BPA.NODE\_KEY | &&

| GROUP BY BP\_ID, COMPANY\_NAME, SO.CURRENCY\_CODE |.

TRY.

"1. create statement object

lo\_stmt = NEW cl\_sql\_statement( ).

"2. execute the query

lo\_res = lo\_stmt->execute\_query( lv\_stmt ).

"3. set the output parameter

lo\_res->set\_param\_table( REF #( lt\_result ) ).

"4. fetch the result

lo\_res->next\_package( ).

"5. release the resources

lo\_res->close( ).

CATCH cx\_sql\_exception INTO DATA(lx).

"do some meaningful error handling

WRITE: lx->sql\_message.

ENDTRY.

cl\_demo\_output=>display\_data( lt\_result ).

## ZR\_AMDP\_01\_SIMPLE\_CALL

REPORT zr\_amdp\_01\_simple\_call.

DATA(lo\_amdp) = NEW zcl\_amdp\_simple\_00( ).

TRY.

lo\_amdp->get\_customer\_infos(

IMPORTING

et\_customer\_info = DATA(lt\_result) ).

cl\_demo\_output=>display\_data( lt\_result ).

CATCH cx\_amdp\_execution\_failed INTO DATA(lx).

"do some meaningful error handling

WRITE: 'Error occurred during execution of an AMDP - do something!',

/ 'SQL Error Message:',

/ |{ lx->sql\_message }|.

ENDTRY.

## ZR\_AMDP\_BADI\_CALL

REPORT zr\_amdp\_badi\_call.

data lt\_cust\_info type zif\_a4h1\_amdp\_badi=>tt\_customer\_info.

"call the AMDP BAdI

zcl\_a4h1\_call\_amdp\_badi=>call\_badi( IMPORTING et\_customer\_info = lt\_cust\_info ).

cl\_demo\_output=>display\_data( name = 'Customer Info' value = lt\_cust\_info ).

## ZR\_AMDP\_CALL

REPORT zr\_amdp\_call.

DATA(lo\_info\_list) = NEW zcl\_demo\_paid\_on\_date\_amdp( ).

lo\_info\_list->paid\_on\_date(

EXPORTING

iv\_payment\_date = '20140912'

IMPORTING

et\_customer\_info = DATA(lt\_cust\_info\_amdp)

et\_invoice\_header = DATA(lt\_inv\_head\_amdp)

et\_invoice\_item = DATA(lt\_inv\_item\_amdp) ).

cl\_demo\_output=>display\_data( name = 'Customer Info'

value = lt\_cust\_info\_amdp ).

# ABAP Interfaces

## ZIF\_A4H1\_AMDP\_BADI

INTERFACE zif\_a4h1\_amdp\_badi

PUBLIC .

INTERFACES if\_badi\_interface .

TYPES:

BEGIN OF ty\_customer\_info,

customer\_id TYPE snwd\_bpa-bp\_id,

customer\_name TYPE snwd\_bpa-company\_name,

currency\_code TYPE snwd\_so-currency\_code,

total\_gross\_amount TYPE snwd\_so-gross\_amount,

END OF ty\_customer\_info ,

tt\_customer\_info TYPE STANDARD TABLE OF ty\_customer\_info.

METHODS get\_customer\_infos

EXPORTING

VALUE(et\_customer\_info) TYPE tt\_customer\_info

RAISING cx\_amdp\_error .

ENDINTERFACE.