Using OOPS ABAP (Class Concept) in Transformations in SAP BI

Applies to:

BI 7.0. For more information, visit the EDW homepage

Summary

This article explains a step by step procedure on how to use OOPS ABAP in BI Transformations with an example.

Author: Hemant Khemani

Company: Wipro Technologies

Created on: 06th January 2011

Author Bio

Hemant Khemani is a SAP BI Consultant with Wipro Technologies. He has more than 4 years of experience in SAP BI.

Table of Contents

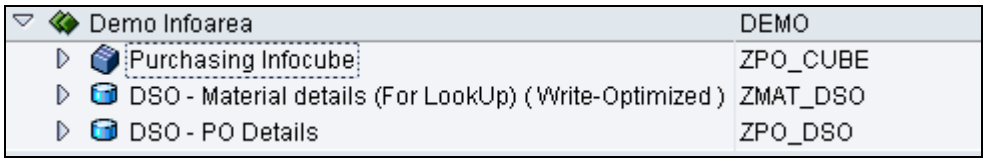
Introduction

Now-a-days, most clients demand that routines in various transformations in SAP BI must be coded using OOPS ABAP methodology. The main advantage of using OOPS ABAP methodology in transformations is given below.

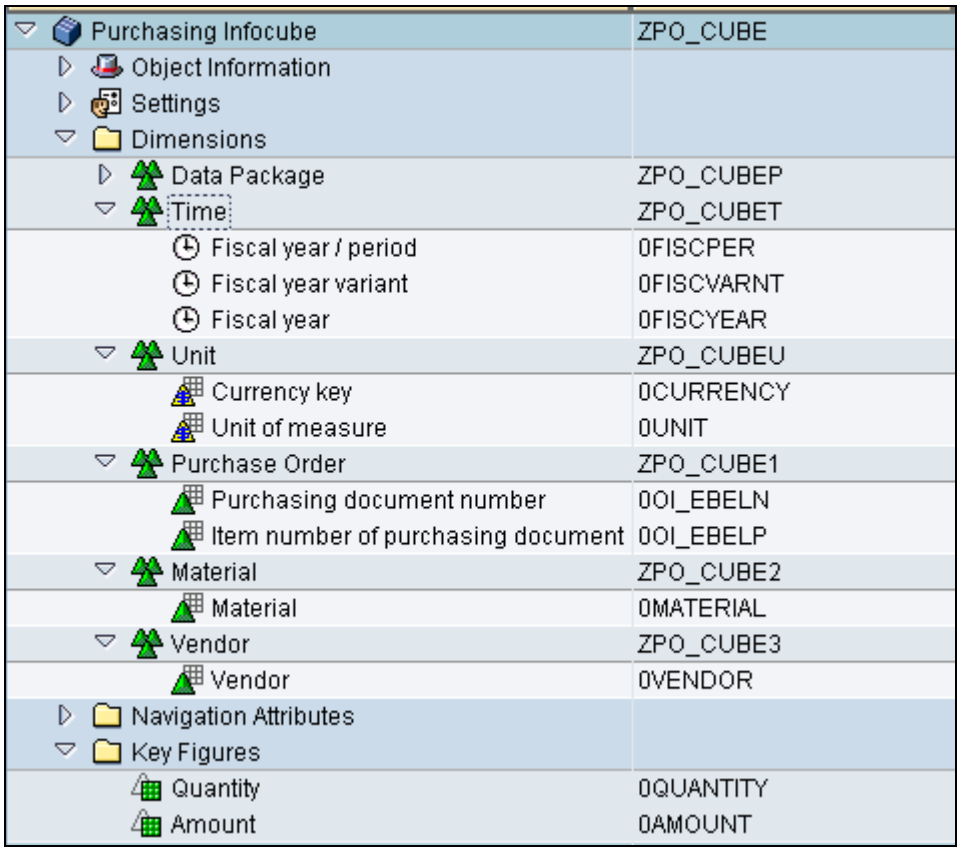
- Similar routine logic need not to be written again. We can create generic method and use it at multiple places.

Scenario

We will be loading purchasing related information to BW. The objects involved in the scenario are listed below:



1. ZPO_CUBE -> Target Cube



2. ZPO_DSO -> Source DSO

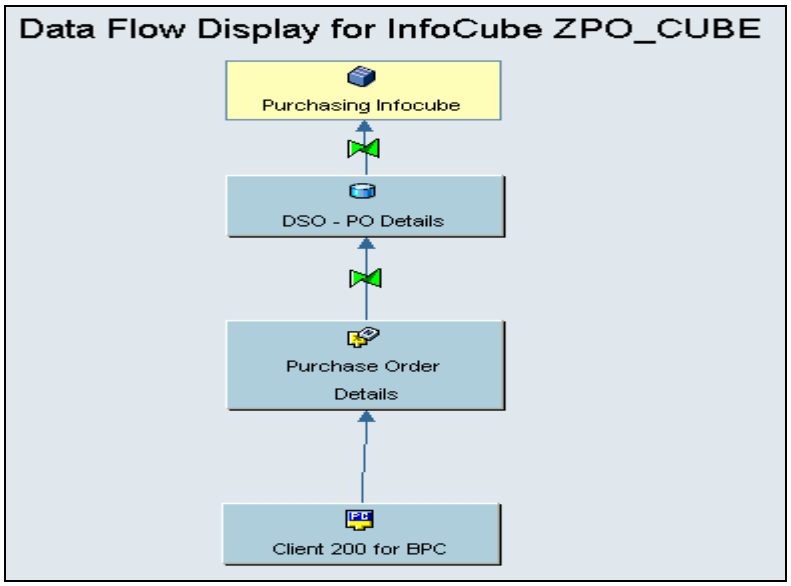
DSO - PO Details	ZPO_DSO
Object Information	
Settings	
Key fields	
Purchasing document number	0OI_EBELN
Item number of purchasing document	0OI_EBELP
Data Fields	
Date on which the record was created	0CREATEDON
Material	0MATERIAL
Quantity	0QUANTITY
Unit of measure	0UNIT
Navigation Attributes	
Indexes	

3. ZMAT_DSO -> LOOKUP DSO

DSO - Material details (For LookUp)	ZMAT_DSO
Object Information	
Settings	
Technical Key (Generated)	
Semantic Key	
Material	0MATERIAL
Data Fields	
Vendor	0VENDOR
Price Per Unit	ZPR_UNIT
Currency key	0CURRENCY
Navigation Attributes	
Indexes	

Dataflow

The purchasing data will be loaded from PO Details DSO ZPO_DSO to purchasing Infocube ZPO_CUBE.

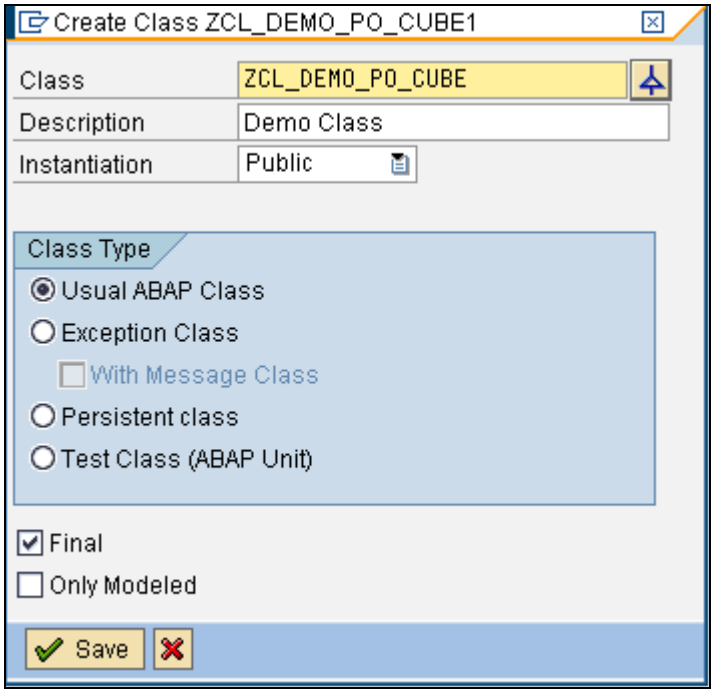


Some of the information (vendor, currency, amount) which is not present in the source DSO ZPO_DSO will be populated by reading data from DSO ZMAT_DSO in the transformation from DSO ZPO_DSO -> Cube ZPO_CUBE.

Step by Step Procedure

Creating Class

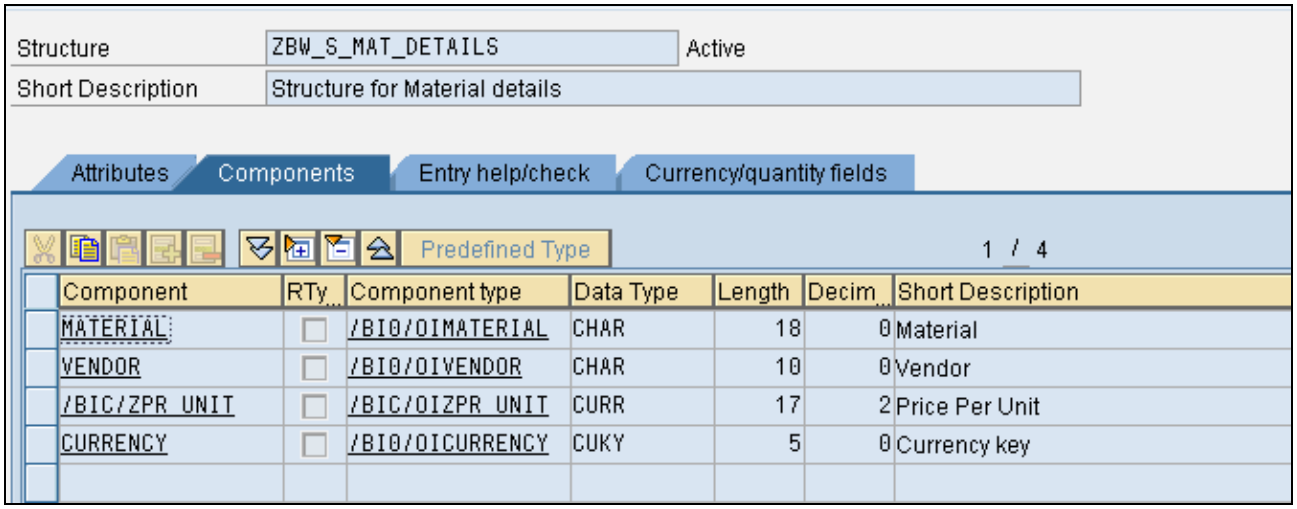
Step 1: Go to SE24 -> Give the Class Name and click on Create.



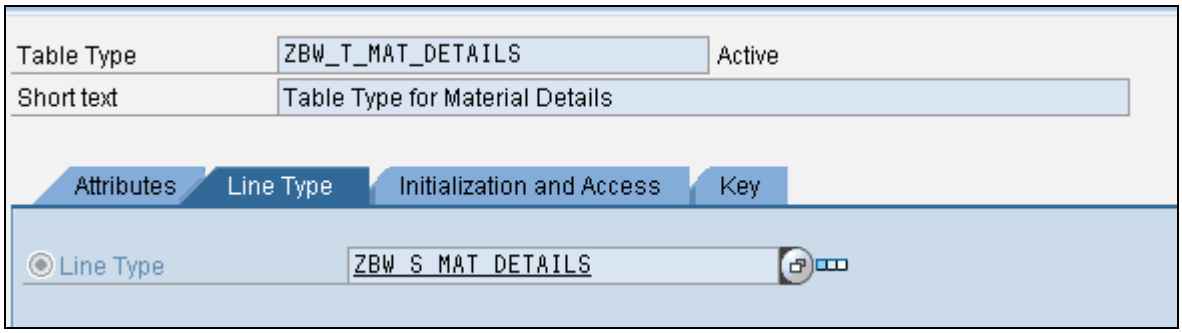
Step 2: Creating Class Attribute

Create an internal table (in our example) as an attribute of the class. For this we will follow steps:

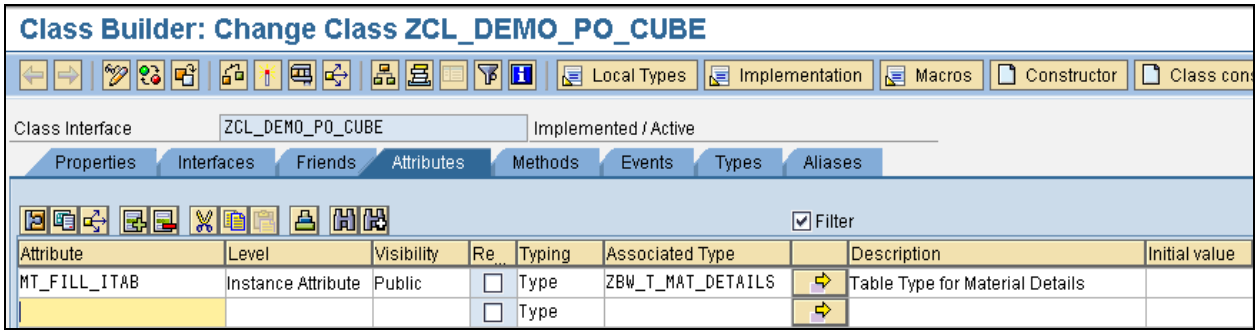
a. Create Structure



b. Create Table Type



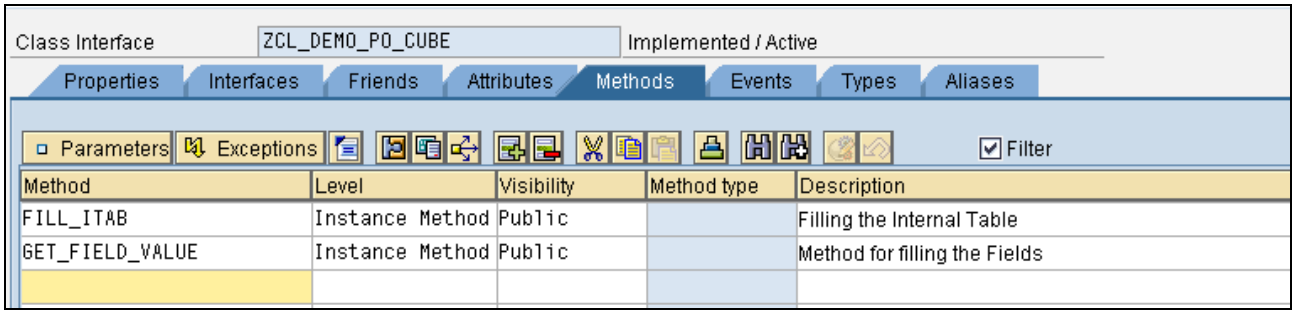
c. Create attribute MT_FILL_ITAB with associated type as ZBW_T_MAT_DETAILS



The above attribute MT_FILL_ITAB can now be used as an internal table in the class methods.

Step 3: Declaring Class Methods

a. Go to Method tab and declare the methods as shown below.



We have used 2 methods as shown above in our example

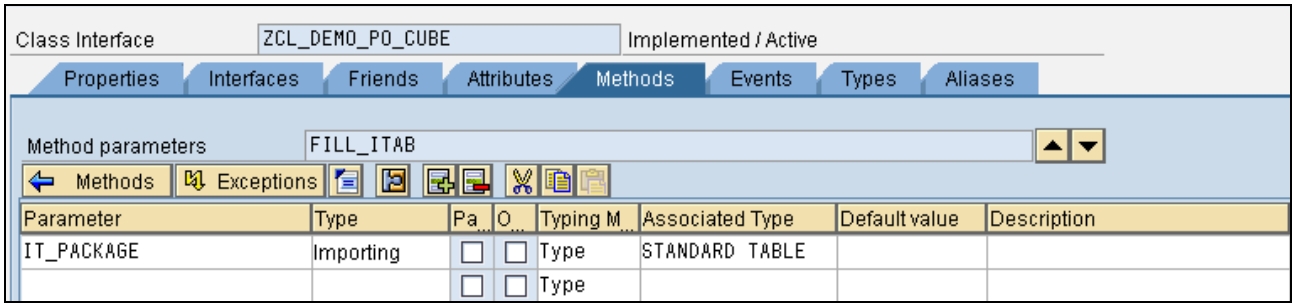
Method 1 - FILL_ITAB

Method 2 - GET_FIELD_VALUE

Step 4: Declaring Method Parameters

Method 1 - FILL_ITAB

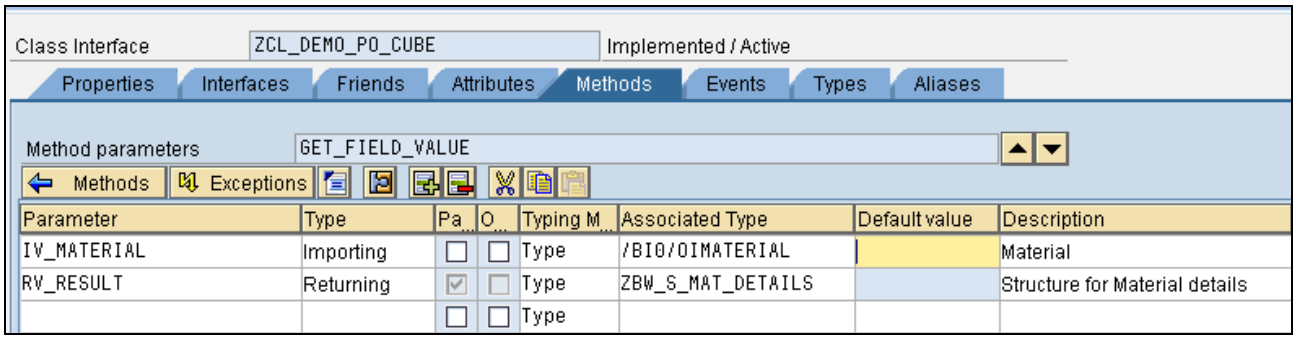
This method will be called from the Start routine in the transformation.



IT_PACKAGE – Importing parameter - is a standard table which will receive the data from the caller.

Method 2 - GET_FIELD_VALUE

This method will be called from field routines in the transformation.



IT_MATERIAL – Importing parameter – this will receive material value from the caller.

RV_RESULT (structure) – Returning parameter – this will send required data to the caller.

Step 5: Declaring Exceptions (if any)

We are not using any exception for the methods in our example.

Step 6: Defining Class Methods

a. Method FILL_ITAB --- In this method we will read the required data from the Look up DSO ZMAT_DSO and fill it into the internal table MT_FILL_ITAB.

Method FILL_ITAB Active

```
METHOD FILL_ITAB.

  DATA: LT_WHERE TYPE STANDARD TABLE OF STRING,
        LV_WHERE TYPE STRING.

  REFRESH: MT_FILL_ITAB.

  CHECK LINES( IT_PACKAGE ) > 0.

  CONCATENATE 'MATERIAL = ' ' IT_PACKAGE-MATERIAL' INTO LV_WHERE.
  APPEND LV_WHERE TO LT_WHERE.

  SELECT MATERIAL VENDOR /BIC/ZPR_UNIT CURRENCY
  FROM /BIC/AZMAT_DSO00
  INTO CORRESPONDING FIELDS OF TABLE MT_FILL_ITAB
  FOR ALL ENTRIES IN IT_PACKAGE
  WHERE (LT_WHERE).

ENDMETHOD.
```

b. Method GET_FIELD_VALUE --- In this method we will read the required data based on the material value from the internal table MT_FILL_ITAB and will send the corresponding details back to the caller in structure RV_RESULT.

Method GET_FIELD_VALUE Active

```
METHOD GET_FIELD_VALUE.

  DATA: WA_FILL_ITAB LIKE LINE OF MT_FILL_ITAB.

   CLEAR: WA_FILL_ITAB, RV_RESULT.

* If the record is found: Return the values from ZP0D0006.
   READ TABLE MT_FILL_ITAB INTO WA_FILL_ITAB WITH KEY
         MATERIAL  = IV_MATERIAL.

   IF SY-SUBRC = 0.
     RV_RESULT = WA_FILL_ITAB.
   ENDIF.

ENDMETHOD.
```

Step 7: Save and Activate the Class.

Using the Class in Transformation

Global Declaration

In start routine we will first declare the class under global declaration so that we can use the class in the start routine and field routines.

```
*$*$ begin of global - insert your declaration only below this line  *-*
    ... "insert your code here
    DATA: MR_TRANSFORMATION TYPE REF TO ZCL_DEMO_PO_CUBE.
*$*$ end of global - insert your declaration only before this line   *-*
```

Global data type to receive values returned from the method GET_FIELD_VALUE.

```
  *$*$ begin of 2nd part global - insert your code only below this line  *
  ... "insert your code here
  DATA lv_get_field type ZBW_S_MAT_DETAILS.
  *$*$ end of 2nd part global - insert your code only before this line   *
```

Creating class object

```
    IF MR_TRANSFORMATION IS NOT BOUND.
      CREATE OBJECT MR_TRANSFORMATION.
    ENDIF.
```

Calling Methods

a. Method 1 – FILL ITAB

This method is called in the start routine.

As we can see below, SOURCE_PACKAGE is passed to this method. The method then populates the internal table with the required information from DSO ZMAT_DSO.

```
    MR_TRANSFORMATION->FILL_ITAB( SOURCE_PACKAGE ).
```

b. Method 2 – GET_FIELD_VALUE

This method is called in the field routines for 0VENDOR and 0AMOUNT.

The input passed to the method is 0MATERIAL.

For 0VENDOR

```
    LV_GET_FIELD = MR_TRANSFORMATION->GET_FIELD_VALUE(
    IV_MATERIAL = SOURCE_FIELDS-MATERIAL ).

    RESULT = LV_GET_FIELD-VENDOR.
```

For 0AMOUNT

```
    LV_GET_FIELD = MR_TRANSFORMATION->GET_FIELD_VALUE(
    IV_MATERIAL = SOURCE_FIELDS-MATERIAL ).

    RESULT = LV_GET_FIELD-/BIC/ZPR_UNIT * SOURCE_FIELDS-QUANTITY.
    CURRENCY = LV_GET_FIELD-CURRENCY.
```

Next, we will run the DTP to load the data.

Checking the Data

Source Data – DSO ZPO_DSO

Data Browser: Table /BIC/AZPO_DSO00 Select Entries 11

OI_EBELN	OI_EBELP	CREATEDON	MATERIAL	QUANTITY	UNIT	RECORDMODE
4500000001	2	09.12.2010	000000000000047321	100,000	ST	
4500000012	1	16.12.2010	000000000000004675	50,000	KG	
4500000041	1	11.08.2010	000000000000000123	120,000	EA	
4500000051	4	25.11.2009	000000000000000658	75,000	ST	
4500000072	1	18.03.2010	000000000000047321	100,000	ST	
4500000093	2	17.09.2009	000000000000987953	90,000	KG	
4506501271	1	26.01.2009	000000000000000123	200,000	EA	
4511123189	2	31.01.2010	000000000000987953	10,000	KG	
4658122795	1	01.12.2010	000000000000000123	65,000	EA	
4700124879	3	25.10.2010	000000000000021732	500,000	KG	
5500001242	1	20.05.2010	000000000000000658	20,000	ST	

LOOKUP Data – DSO ZMAT_DSO

Data Browser: Table /BIC/AZMAT_DSO00 Select Entries 7

REQUEST	D...	R...	MATERIAL	VENDOR	/BIC/ZPR_UNIT	CURRENCY	RECORDMODE
DTPR_D9HD9B99WAWZ56GBFLEZUM6D4	1	1	000000000000000123	0000004566	500,00	USD	
DTPR_D9HD9B99WAWZ56GBFLEZUM6D4	1	2	000000000000000658	0000001278	65,00	EUR	
DTPR_D9HD9B99WAWZ56GBFLEZUM6D4	1	3	000000000000004675	0000000189	20,00	EUR	
DTPR_D9HD9B99WAWZ56GBFLEZUM6D4	1	4	000000000000021732	0000000014	50,00	USD	
DTPR_D9HD9B99WAWZ56GBFLEZUM6D4	1	5	000000000000047321	0000007892	200,00	EUR	
DTPR_D9HD9B99WAWZ56GBFLEZUM6D4	1	6	000000000000219132	0000056233	100,00	USD	
DTPR_D9HD9B99WAWZ56GBFLEZUM6D4	1	7	000000000000987953	0000000187	10,00	EUR	

Target data – Infocube ZPO_CUBE

"ZPO_CUBE", List output

0OI_EBELN	Ite...	0MATERIAL	0VENDOR	0FISCPER	0FISCVARNT	Fiscal ye...	Currency	0UNIT	0AMOUNT	Quantity
4500000001	2	000000000000047321	0000007892	2010012	K4	2010	EUR	ST	20.000,00	100
4500000012	1	000000000000004675	0000000189	2010012	K4	2010	EUR	KG	1.000,00	50
4500000041	1	000000000000000123	0000004566	2010008	K4	2010	USD	EA	60.000,00	120
4500000051	4	000000000000000658	0000001278	2009011	K4	2009	EUR	ST	4.875,00	75
4500000072	1	000000000000047321	0000007892	2010003	K4	2010	EUR	ST	20.000,00	100
4500000093	2	000000000000987953	0000000187	2009009	K4	2009	EUR	KG	900,00	90
4506501271	1	000000000000000123	0000004566	2009001	K4	2009	USD	EA	100.000,00	200
4511123189	2	000000000000987953	0000000187	2010001	K4	2010	EUR	KG	100,00	10
4658122795	1	000000000000000123	0000004566	2010012	K4	2010	USD	EA	32.500,00	65
4700124879	3	000000000000021732	0000000014	2010010	K4	2010	USD	KG	25.000,00	500
5500001242	1	000000000000000658	0000001278	2010005	K4	2010	EUR	ST	1.300,00	20

Related Content

SAP help on Routines in Transformations

Beginner's Guide to BI - ABAP

For more information, visit the EDW homepage

Disclaimer and Liability Notice

This document may discuss sample coding or other information that does not include SAP official interfaces and therefore is not supported by SAP. Changes made based on this information are not supported and can be overwritten during an upgrade.

SAP will not be held liable for any damages caused by using or misusing the information, code or methods suggested in this document, and anyone using these methods does so at his/her own risk.

SAP offers no guarantees and assumes no responsibility or liability of any type with respect to the content of this technical article or code sample, including any liability resulting from incompatibility between the content within this document and the materials and services offered by SAP. You agree that you will not hold, or seek to hold, SAP responsible or liable with respect to the content of this document.