Downloading InfoProvider Metadata to Excel

Applies to:

SAP BW 3.5, SAP BI 7.0 etc. For more information, visit the EDW homepage.

Summary

This paper has an approach to download the Metadata of an InfoProvider to Excel sheet.

Author: Poornima Gayatri

Company: Deloitte Consulting

Created on: 22 March 2011

Author Bio

Poornima Gayatri Chennur is currently working in Deloitte Consulting India Pvt. Ltd. She is working on SAP BW/BI from last 3.7 years

Table of Contents

Introduction

During the technical documentation of project, we need to prepare documentations on DSO, Cube, MultiProvider, InfoSet, InfoObject etc. and list their metadata in an excel sheet. It becomes very difficult for us to copy and paste each and every field of metadata.

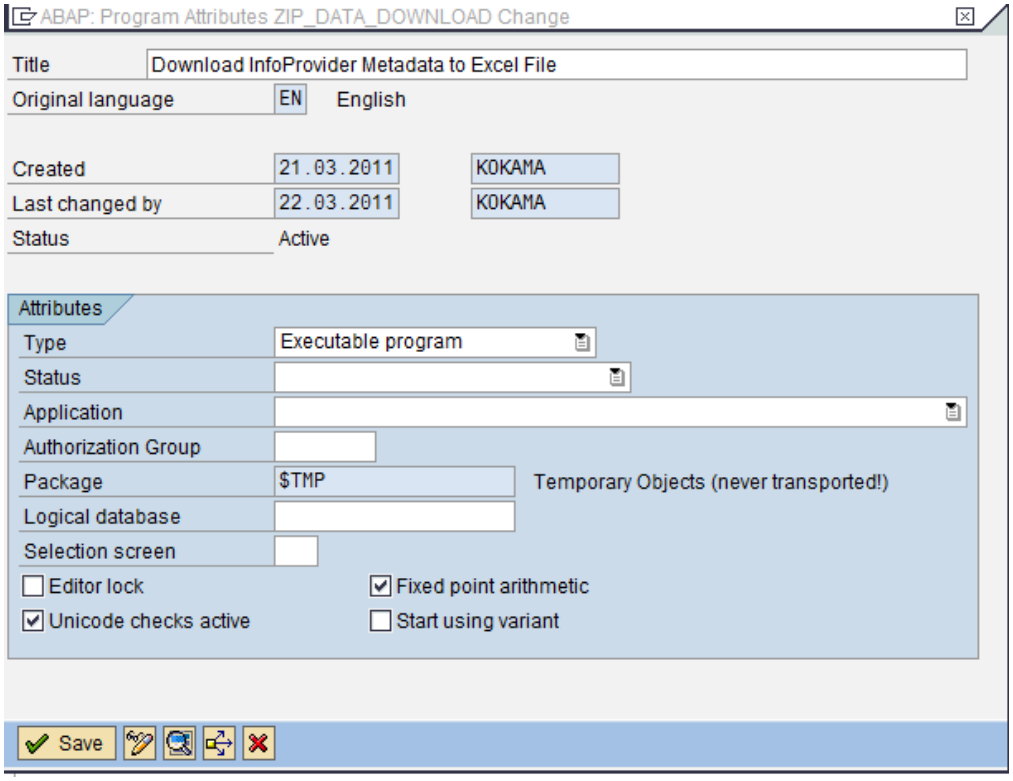
This document has a program which can download the Metadata of InfoProvider into Excel sheet.

Scenario

To demonstrate the functionality and usage we have created one custom Transaction code 'ZIP_DOWNLOAD'. The transaction code will be run on the program 'ZIP_DATA_DOWNLOAD' which has logic to download the contents of InfoProvider to Excel file.

Step 1: Creating Custom Program

In Transaction code SE38 (ABAP EDITOR), create a new program with name 'ZIP_DATA_DOWNLOAD' and the attributes of the program.



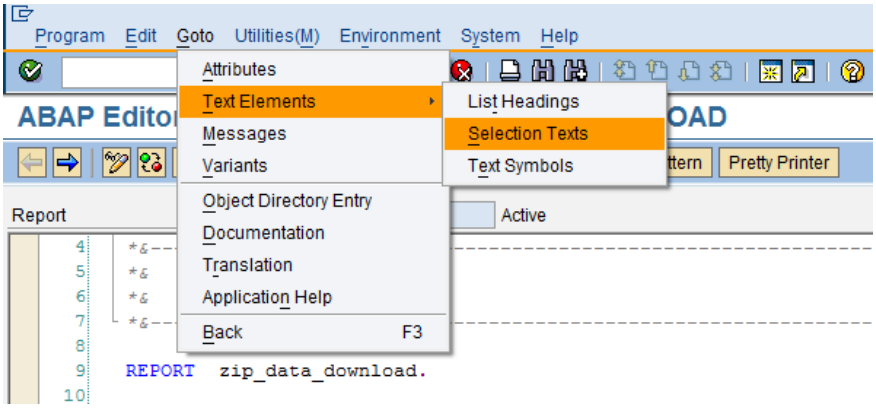
Please copy the Source code for the above program under Source code section. Once program copied, save the program and activate.

Step 2: Settings for Selection Texts

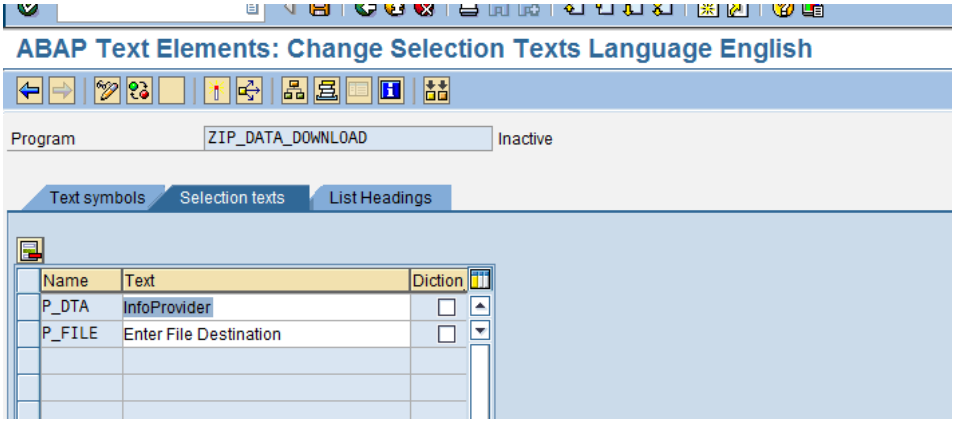
To display user defined Selection texts at input selection screen after program execution, we need to define Selection texts for the entries defined in the program.

Go to the program editor window in SE38 and select

Goto (Menu) -> Text Elements -> Selection Texts.



Enter the below entries in 'Selection Texts' tab.

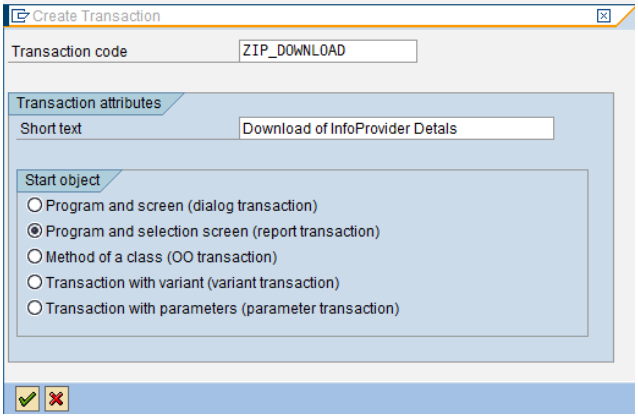


Step 3: Creating ZIP_DOWNLOAD Custom Transaction Code

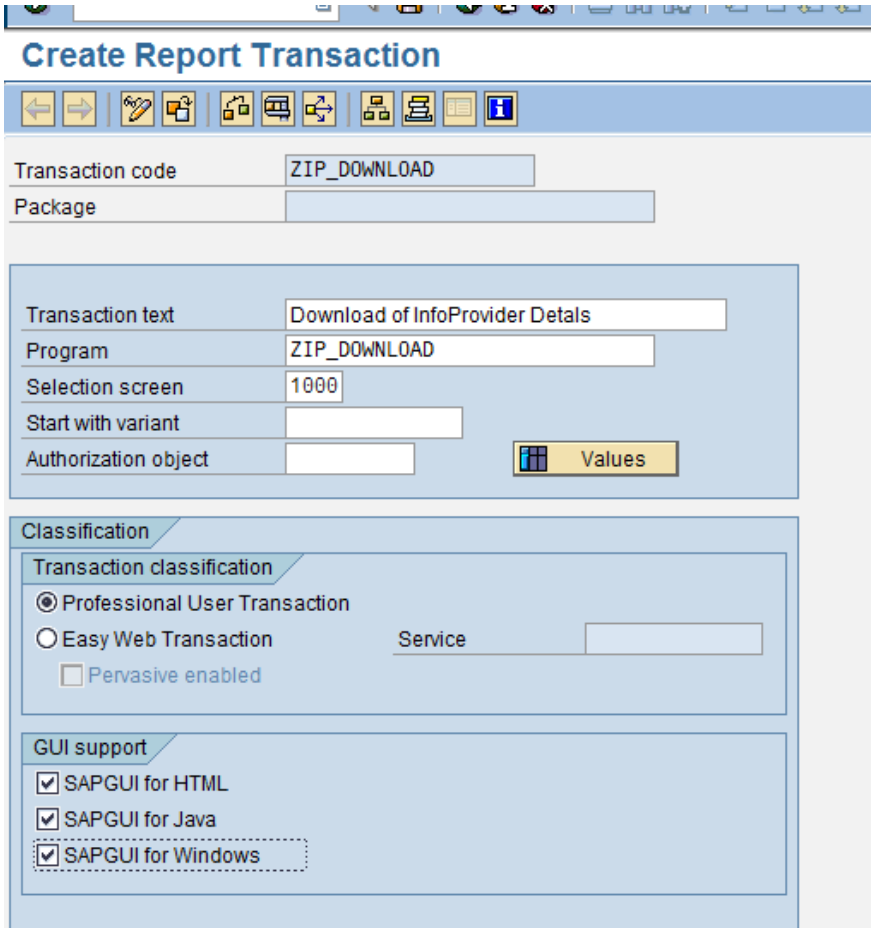
To call and execute 'ZIP_DATA_DOWNLOAD' custom program through transaction code we need to create new 'Z' transaction code with name 'ZIP_DOWNLOAD'. The following steps to be followed to create ZRSZC transaction code.

Go to SE93 transaction code to create custom transaction code.

Type 'ZIP_DOWNLOAD' and press 'Create' button as shown below



Then it will display another screen shown below and enter all given parameters (including the program name (ZIP_DOWNLOAD) we have created) with required check options.



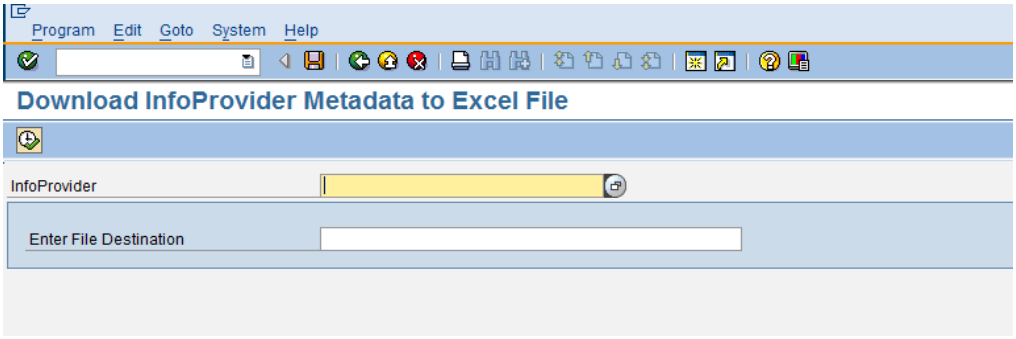
After entering required parameters press ‘Save’ button to save the transaction code.

Step 4: Results

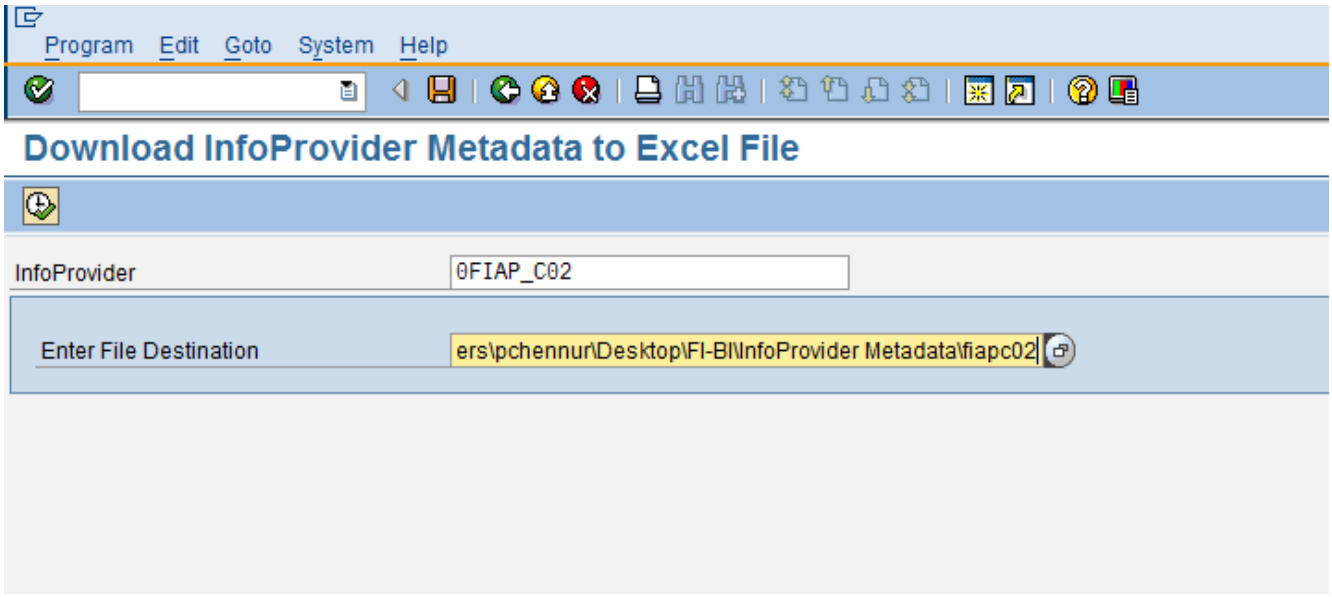
To see the results we have to follow the below instructions.

Open new SAP screen and enter ZIP_DOWNLOAD transaction code in the command text box as shown below.

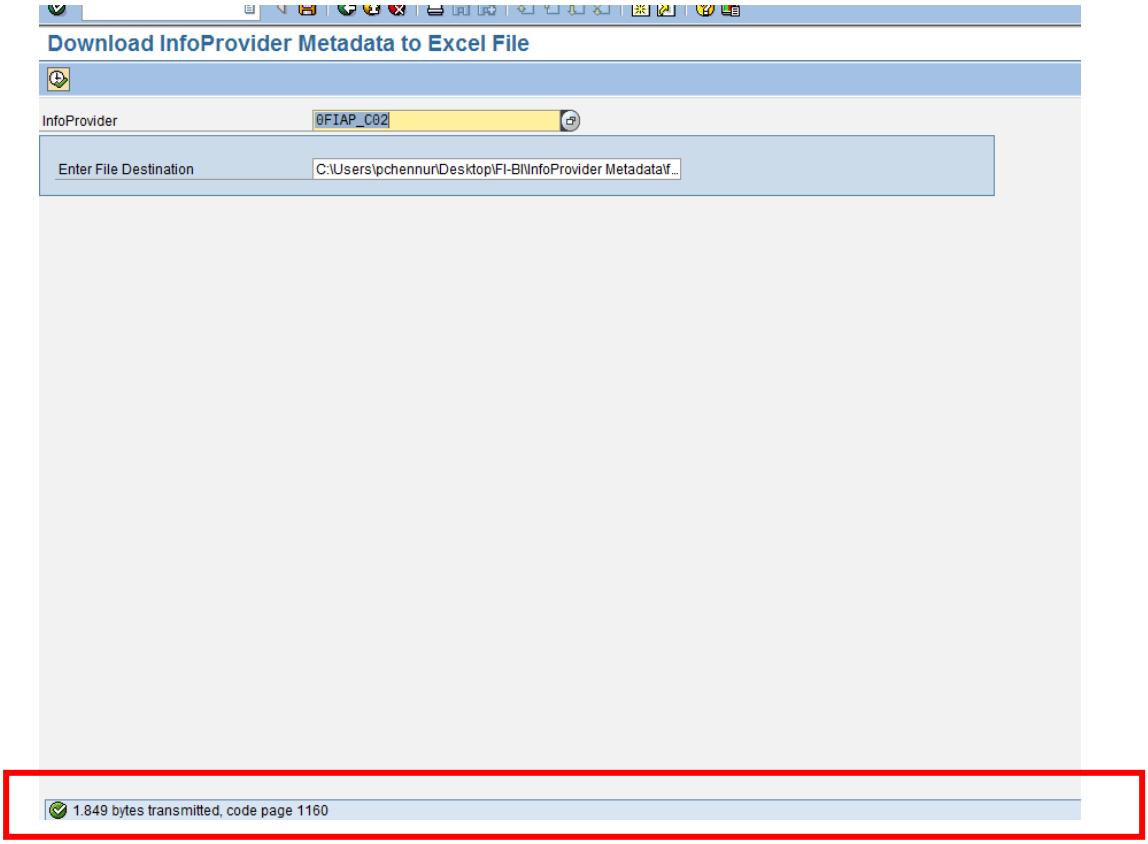
Once the above transaction code is run, it will execute the assigned ABAP program at back-end and display the below input selection screen.



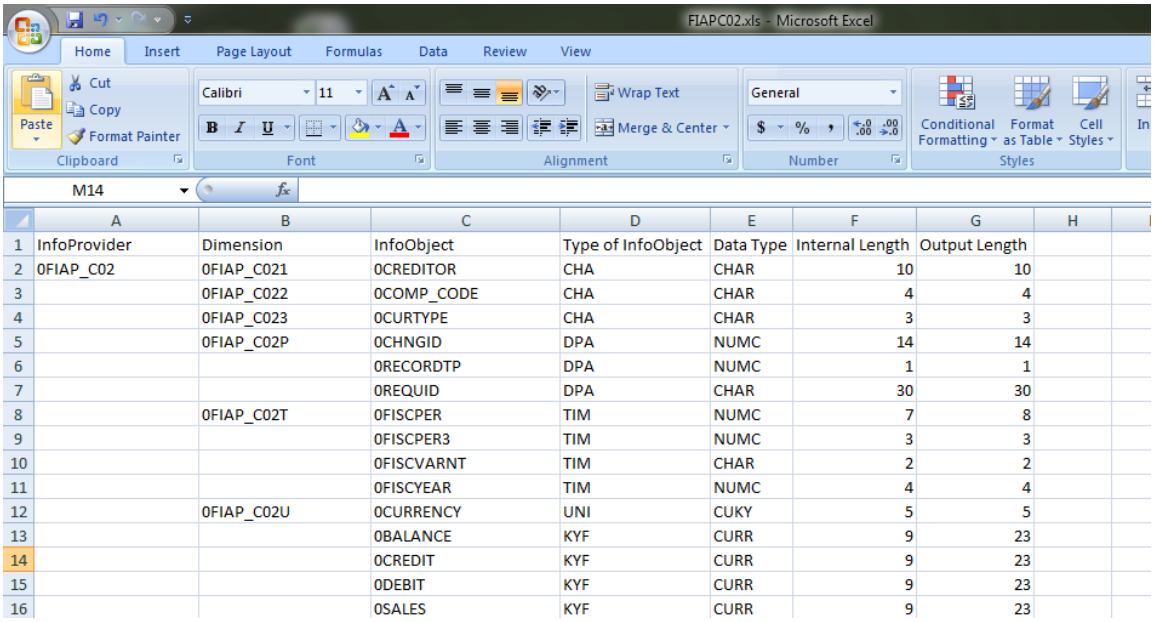
Enter the name of the InfoProvider, from which you require the Metadata, and the destination of the file to be downloaded.



After execution we can see the message saying that the file is downloaded.



Excel File with InfoProvider Metadata



	A	B	C	D	E	F	G	H
1	InfoProvider	Dimension	InfoObject	Type of InfoObject	Data Type	Internal Length	Output Length	
2	0FIAP_C02	0FIAP_C021	0CREDITOR	CHA	CHAR	10	10	
3		0FIAP_C022	0COMP_CODE	CHA	CHAR	4	4	
4		0FIAP_C023	0CURTYPE	CHA	CHAR	3	3	
5		0FIAP_C02P	0CHNGID	DPA	NUMC	14	14	
6			0RECORDTP	DPA	NUMC	1	1	
7			0REQUID	DPA	CHAR	30	30	
8		0FIAP_C02T	0FISCPER	TIM	NUMC	7	8	
9			0FISCPER3	TIM	NUMC	3	3	
10			0FISCVARNT	TIM	CHAR	2	2	
11			0FISCYEAR	TIM	NUMC	4	4	
12		0FIAP_C02U	0CURRENCY	UNI	CUKY	5	5	
13			0BALANCE	KYF	CURR	9	23	
14			0CREDIT	KYF	CURR	9	23	
15			0DEBIT	KYF	CURR	9	23	
16			0SALES	KYF	CURR	9	23	

Report Source Code

```
*&---------------------------------------------------------------------*
*& Report zip_data_download
*&
*&---------------------------------------------------------------------*
*&
*&
*&---------------------------------------------------------------------*

REPORT  zip_data_download.

TYPE-POOLS: rsd, rsdq.

DATA:
  g_s_dta          TYPE rsd_s_dta,
  g_t_ioinf        TYPE rsdq_t_iobj_info,
  g_t_dta_dime     TYPE rsd_t_dta_dime.
PARAMETER:
  p_dta            TYPE rsinfoprov        OBLIGATORY
                                          MEMORY ID /bic/rsdq/infoprov.
SET PARAMETER ID '/BIC/RSDQ/INFOPROV'   FIELD p_dta.

DATA: lo_gui TYPE REF TO cl_gui_frontend_services,
lv_title TYPE string,
lv_folder TYPE string,
lv_dir TYPE string,
v_filetype TYPE string,
v_filename TYPE string,
lv_filename TYPE string.

SELECTION-SCREEN BEGIN OF BLOCK screen.
PARAMETER p_file LIKE rlgrap-filename.
SELECTION-SCREEN END OF BLOCK screen.
```

```
AT SELECTION-SCREEN ON VALUE-REQUEST FOR p_file.
  PERFORM f_browse CHANGING p_file.

START-OF-SELECTION.

  PERFORM z_convert_excel.

*&---------------------------------------------------------------------*
*&      Form  f_browse
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
*      -->FC_FILE    text
*----------------------------------------------------------------------*
FORM f_browse CHANGING fc_file.
  DATA: lo_gui TYPE REF TO cl_gui_frontend_services,
  lv_title TYPE string,
  lv_folder TYPE string,
  lv_dir TYPE string.

  CREATE OBJECT lo_gui.
  lv_title = 'Download of InfoProvider Data to Excel'.
  lv_folder = 'C:'.
  CALL METHOD lo_gui->directory_browse
    EXPORTING
      window_title    = lv_title
      initial_folder  = lv_folder
    CHANGING
      selected_folder = lv_dir.
  fc_file = lv_dir.

ENDFORM.                    "f_browse

*&---------------------------------------------------------------------*
*&      Form  Z_CONVERT_EXCEL
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
FORM z_convert_excel .

  TYPES : BEGIN OF l_infocube,
          infocube TYPE rsinfocube,
          END OF l_infocube.
  DATA : lt_infocube TYPE TABLE OF l_infocube,
         wa_infocube TYPE l_infocube.


  SELECT infocube FROM rsdcube INTO TABLE lt_infocube WHERE infocube =  p_dta .


  v_filetype = '.xls'.
  v_filename = p_file.
  CONCATENATE p_file v_filetype INTO lv_filename.
```

```
CALL FUNCTION 'RSDQ_GET_DTA_INFO'
  EXPORTING
    i_infoprov   = p_dta
  IMPORTING
    e_s_dta      = g_s_dta
    e_t_ioinf    = g_t_ioinf
    e_t_dta_dime = g_t_dta_dime
  EXCEPTIONS
    OTHERS       = 1.
IF sy-subrc <> 0.
  MESSAGE e203(dbman) WITH p_dta.
ENDIF.

TYPES : BEGIN OF l_data,
        infoprov  TYPE rsinfoprov,   "InfoProvider
        dimension TYPE rsdimension,  " Dimension
        iobjnm    TYPE rsiobjnm,     "InfoObject
        iobjtp    TYPE rsiobjtp,         "Type of an InfoObject
        datatp    TYPE datatype_d,   "Data Type in ABAP Dictionary
        intlen    TYPE rsdintlen,    "Internal Length of InfoObjects in BW
        outputlen TYPE outputlen,    "Output Length

  END OF l_data.


DATA : wa_inf TYPE rsdq_s_iobj_info,
       wa_inf1 TYPE l_data,
       wa_inf2 TYPE l_data,
       l_inf TYPE STANDARD TABLE OF l_data,
       lv_inf1 TYPE rsinfoprov,
       lv_inf2 TYPE rsdimension.

CLEAR : lv_inf2.

IF lt_infocube IS NOT INITIAL.

  DATA : BEGIN OF int_head OCCURS 0,
filed1(20) TYPE c,                     " Header Data
END OF int_head.

  int_head-filed1 = 'InfoProvider'.
  APPEND int_head.
  CLEAR int_head.

  int_head-filed1 = 'Dimension'.
  APPEND int_head.
  CLEAR int_head.

  int_head-filed1 = 'InfoObject'.
  APPEND int_head.
  CLEAR int_head.

  int_head-filed1 = 'Type of InfoObject'.
  APPEND int_head.
  CLEAR int_head.
```

```
int_head-filed1 = 'Data Type'.
APPEND int_head.
CLEAR int_head.

int_head-filed1 = 'Internal Length'.
APPEND int_head.
CLEAR int_head.

int_head-filed1 = 'Output Length'.
APPEND int_head.
CLEAR int_head.
LOOP AT g_t_ioinf INTO wa_inf.

 CLEAR wa_inf1.

  IF sy-tabix > 1.
    CLEAR wa_inf1-infoprov.
  ELSE.
    wa_inf1-infoprov = wa_inf-infoprov.
  ENDIF.

  IF wa_inf-dimension NE lv_inf2.
    wa_inf1-dimension = wa_inf-dimension.
    lv_inf2 = wa_inf-dimension.
  ENDIF.

  wa_inf1-iobjnm = wa_inf-iobjnm.
  wa_inf1-iobjtp = wa_inf-iobjtp.
  wa_inf1-datatp = wa_inf-datatp.
  wa_inf1-intlen  = wa_inf-intlen.
  wa_inf1-outputlen = wa_inf-outputlen.
  APPEND wa_inf1 TO l_inf.

  CLEAR : wa_inf.
ENDLOOP.

CALL FUNCTION 'GUI_DOWNLOAD'
  EXPORTING
    filename                = lv_filename
    filetype                = 'ASC'
    write_field_separator   = 'X'

  TABLES
    data_tab                = l_inf
    fieldnames              = int_head
  EXCEPTIONS
    file_write_error        = 1
    no_batch                = 2
    gui_refuse_filetransfer = 3
    invalid_type            = 4
    no_authority            = 5
    unknown_error           = 6
    header_not_allowed      = 7
    separator_not_allowed   = 8
    filesize_not_allowed    = 9
    header_too_long         = 10
```

```
      dp_error_create         = 11
      dp_error_send           = 12
      dp_error_write          = 13
      unknown_dp_error        = 14
      access_denied           = 15
      dp_out_of_memory        = 16
      disk_full               = 17
      dp_timeout              = 18
      file_not_found          = 19
      dataprovider_exception  = 20
      control_flush_error     = 21
      OTHERS                  = 22.

  IF sy-subrc <> 0.

  ENDIF.

ELSE.

  DATA : BEGIN OF int_head1 OCCURS 0,
 filed1(20) TYPE c,                    " Header Data
 END OF int_head1.

  int_head1-filed1 = 'InfoProvider'.
  APPEND int_head1.
  CLEAR int_head1.

  int_head1-filed1 = 'Key/Data Field'.
  APPEND int_head1.
  CLEAR int_head1.

  int_head1-filed1 = 'InfoObject'.
  APPEND int_head1.
  CLEAR int_head1.

  int_head1-filed1 = 'Type of InfoObject'.
  APPEND int_head1.
  CLEAR int_head1.

  int_head1-filed1 = 'Data Type'.
  APPEND int_head1.
  CLEAR int_head1.

  int_head1-filed1 = 'Internal Length'.
  APPEND int_head1.
  CLEAR int_head1.

  int_head1-filed1 = 'Output Length'.
  APPEND int_head1.
  CLEAR int_head1.

  LOOP AT g_t_ioinf INTO wa_inf.
    CLEAR wa_inf1.

    IF sy-tabix > 1.
      CLEAR wa_inf1-infoprov.
```

```
      ELSE.
        wa_inf1-infoprov = wa_inf-infoprov.
      ENDIF.

      IF wa_inf-dimension NE lv_inf2.
        wa_inf1-dimension = wa_inf-dimension.
        lv_inf2 = wa_inf-dimension.
      ENDIF.

      wa_inf1-iobjnm = wa_inf-iobjnm.
      wa_inf1-iobjtp = wa_inf-iobjtp.
      wa_inf1-datatp = wa_inf-datatp.
      wa_inf1-intlen  = wa_inf-intlen.
      wa_inf1-outputlen = wa_inf-outputlen.
      APPEND wa_inf1 TO l_inf.

      CLEAR : wa_inf.
    ENDLOOP.

    CALL FUNCTION 'GUI_DOWNLOAD'
      EXPORTING
        filename                = lv_filename
        filetype                = 'ASC'
        write_field_separator   = 'X'
               TABLES
        data_tab                = l_inf
        fieldnames              = int_head1
      EXCEPTIONS
        file_write_error        = 1
        no_batch                = 2
        gui_refuse_filetransfer = 3
        invalid_type            = 4
        no_authority            = 5
        unknown_error           = 6
        header_not_allowed      = 7
        separator_not_allowed   = 8
        filesize_not_allowed    = 9
        header_too_long         = 10
        dp_error_create         = 11
        dp_error_send           = 12
        dp_error_write          = 13
        unknown_dp_error        = 14
        access_denied           = 15
        dp_out_of_memory        = 16
        disk_full               = 17
        dp_timeout              = 18
        file_not_found          = 19
        dataprovider_exception  = 20
        control_flush_error     = 21
        OTHERS                  = 22.
    IF sy-subrc <> 0.
    ENDIF.
  ENDIF.
ENDFORM.
```

Related Content

For more information, visit the Business Intelligence Home Page

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.