How to activate or de-activate Dataslice with a BPS Exit Function

Applies to:

SAP NW2004 (BW 3.5) and following.

Summary

This article explains how you can activate or de-activate BPS Dataslices with a BPS function of type Exit. This kind of functionality is not available as a standard (Dataslices are generally managed from BPS0): with this solution you can embed Dataslice activation or de-activation in Planning Sequence or associate such operations to buttons in Planning Applications, since the fact that provides also an update in BPS buffer, about the Dataslice changed status.

Author: Gianfranco Vallese
Company: BGP Management Consulting S.p.A.
Created on: 05 August 2008

Author Bio

Gianfranco Vallese is Project Leader at BGP Management Consulting S.p.A.
Main areas of expertise: Financial and Management Planning, Consolidation and Reporting.

Table of Contents

Business Scenario

Dataslices are used in BPS to protect data against undesired changes: generally are used to protect “old” data. You can also use them to lock data against further changes during Planning Process: in this kind of implementation you must consider the need to switch dataslices to active or inactive status depending on the planning phase you are working on.

A good solution to is using BPS Variables in Dataslice definition to provide the desired flexibility: For example, changing the values of a BPS Variable, included in a dataslice, you can cause dataslice to protect data in a different version from the one users are working on. However this can be complex because of the number of Dataslices that must be managed in all the process phases. In this case you work on the “logical” status of Dataslices, because the are always active, but changed to protect (or not) data against changes.

An alternative approach in working on the “physical” status of the dataslice..

Note: the solution provided here act on the status and has the main advantage in the fact that aligns the buffer definition of the Dataslice.

Basically the status of Dataslices is stored in table UPC_DATASLICE (field INACTIVE): when you start a Planning Application (e.g. Web Interface) BPS loads the buffer also with the Dataslice status defined in the DB. Changes made in table UPC_DATASLICE are ignored in the buffer, until you re-start the Planning Application In fact, if you try to change the Dataslice status to inactive in the DB table UPC_DATASLICE, functions that play on the data partitions protected by the Dataslice will report errors, because the Dataslice definition in the BPS buffer is active. Vice versa, if you activate one Dataslice status in the DB table UPC_DATASLICE and expect it to prevent changes you will discover that data underlying the Dataslice are modified, because the Dataslice definition in the BPS buffer is active.

Step By Step

In the following part of this paragraph we will explain the solution providing, for each sep, the detailed description of the activities:

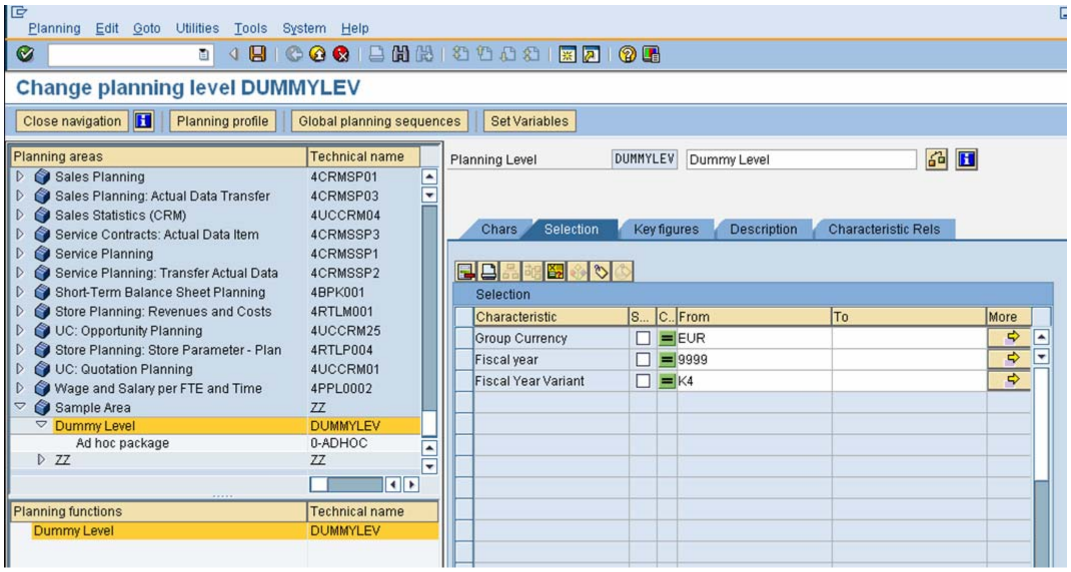
- Create dummy Planning Level
- Create Planning Function of Type EXIT
- Create Parameter Groups

Sample code used to implement FMs belonging to BPS Exit Function is provided in APPENDIX I

Create dummy Planning Level

The BPS function used to activate or de-activate Dataslice does not process data: therefore can be implemented in a specific Planning Level, created ad-hoc, in order to avoid any kind of locking problem.
Create a Planning Level whose selection are referred to a partition of data that will never be involved ineffective planning activities: for example you can choose a year like “9999” or set another characteristic with a non-sense value.

Note: the Planning Level created here doesn’t need to be in the same Area where the Dataslice is defined. The functionality described here works even on different Areas.



Planning areas	Technical name
Sales Planning	4CRMSP01
Sales Planning: Actual Data Transfer	4CRMSP03
Sales Statistics (CRM)	4UCCRM04
Service Contracts: Actual Data Item	4CRMSSP3
Service Planning	4CRMSSP1
Service Planning: Transfer Actual Data	4CRMSSP2
Short-Term Balance Sheet Planning	4BPK001
Store Planning: Revenues and Costs	4RTLM001
UC: Opportunity Planning	4UCCRM25
Store Planning: Store Parameter - Plan	4RTLP004
UC: Quotation Planning	4UCCRM01
Wage and Salary per FTE and Time	4PPL0002
Sample Area	ZZ
Dummy Level	DUMMYLEV
Ad hoc package	0-ADHOC
ZZ	ZZ

Planning functions	Technical name
Dummy Level	DUMMYLEV

Planning Level DUMMYLEV Dummy Level

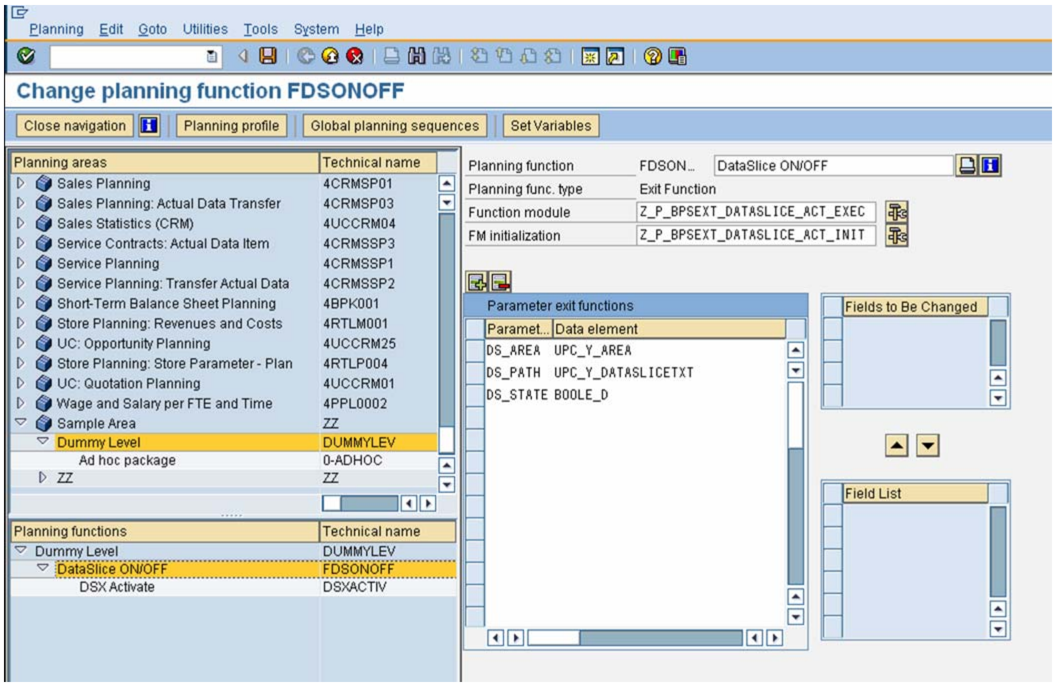
Selection

Characteristic	S...	C..	From	To	More
Group Currency	☐	=	EUR		
Fiscal year	☐	=	9999		
Fiscal Year Variant	☐	=	K4		

Picture 1: Dummy Planning Level

Create Planning Function of Type EXIT

Create a new Planning Function of type “Exit” providing a technical name, and the relative description (e.g. FDSONOFF - “DataSlice ON/OFF”)



Change planning function FDSONOFF

Planning areas	Technical name
Sales Planning	4CRMSP01
Sales Planning: Actual Data Transfer	4CRMSP03
Sales Statistics (CRM)	4UCCRM04
Service Contracts: Actual Data Item	4CRMSSP3
Service Planning	4CRMSSP1
Service Planning: Transfer Actual Data	4CRMSSP2
Short-Term Balance Sheet Planning	4BPK001
Store Planning: Revenues and Costs	4RTLM001
UC: Opportunity Planning	4UCCRM25
Store Planning: Store Parameter - Plan	4RTLP004
UC: Quotation Planning	4UCCRM01
Wage and Salary per FTE and Time	4PPL0002
Sample Area	ZZ
Dummy Level	DUMMYLEV
Ad hoc package	0-ADHOC
ZZ	ZZ

Planning functions	Technical name
Dummy Level	DUMMYLEV
DataSlice ON/OFF	FDSONOFF
DSX Activate	DSXACTIV

Planning function	FDSON...	DataSlice ON/OFF
Planning func. type	Exit Function	
Function module	Z_P_BPSEXT_DATASLICE_ACT_EXEC	
FM initialization	Z_P_BPSEXT_DATASLICE_ACT_INIT	

Parameter exit functions

Paramet...	Data element
DS_AREA	UPC_Y_AREA
DS_PATH	UPC_Y_DATASLICETXT
DS_STATE	BOOLE_D

Fields to Be Changed

Field List

Picture 2: BPS Function of Type EXIT

In the Planning Function definition set the following FMs (the sample code is provided in APPENDIX I):

• FM used for Initialization:	Z_P_BPSEXT_DATASLICE_ACT_INIT
• FM used for Execution:	Z_P_BPSEXT_DATASLICE_ACT_EXEC

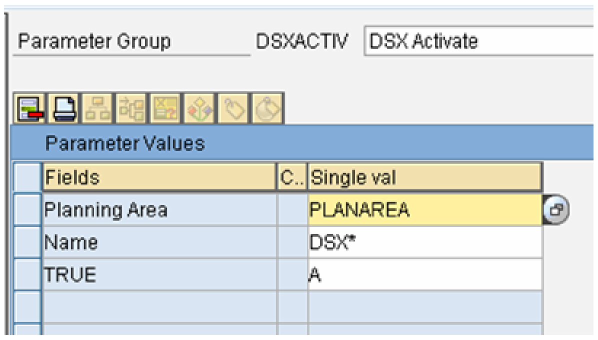
In addition, you must set the following Parameters:

Parameter	Type	Notes
DS_AREA	UPC_Y_AREA	Technical name of the Planning Area where the Dataslice is defined.
DS_PATH	UPC_Y_DATASLICETXT	Name of the DataSlice
DS_STATE	BOOLE_D	Flag use to set Activation or Deactivation of the Dataslice. Valid values are A = Activate and D = De-Activate.

Note: No Characteristic needs to be set in the “fields to be changed”, since the fact that the FMs won’t create or change any transactional data.

Create Parameter Groups

Finally you should create two Parameter Groups, for the specified function: one to set Dataslice Activation and one for De-Activation. As you can see in the following pictures Parameter Group definition will involve 3 different parameter values.
The Planning Area you are requested to specify here can also be a different one from the one we are working in: the BPS Exit Function is able to work on Dataslices located in other Areas.
Both Planning Area and Dataslice name can be typed using wildcards: in the following example all Dataslices whose name starts with “DSX” will be Activated (or De-Activated) in Area “PLANAREA”

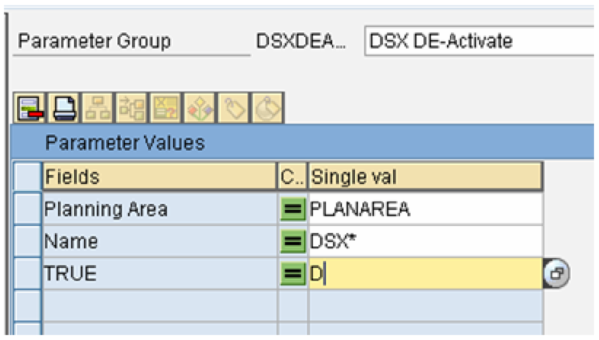


Parameter Group DSXACTIV DSX Activate

Parameter Values

Fields	C..	Single val
Planning Area		PLANAREA
Name		DSX*
TRUE		A

Picture 3: Parameter Group Dataslice Activation



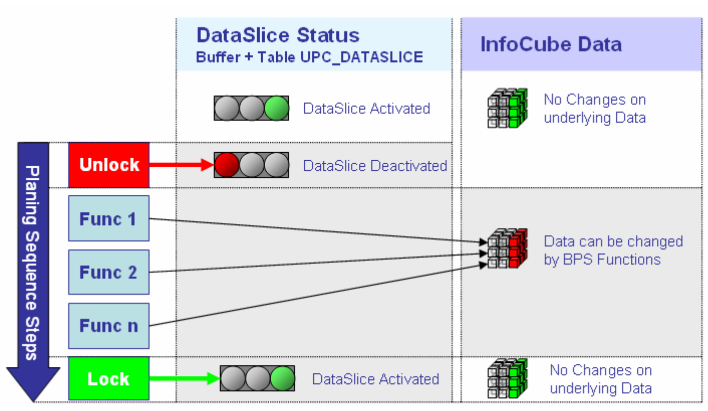
Parameter Group DSXDEA... DSX DE-Activate

Parameter Values

Fields	C..	Single val
Planning Area	=	PLANAREA
Name	=	DSX*
TRUE	=	D

Picture 4: Parameter Group Dataslice De-Activation

Note: planning functions can be included in Planning Sequences (Global) in order to de-activate Dataslice before data manipulation and then to re-activate the dataslice at the end (see the following Picture 5).



Picture 5: Usage in Planning Sequence

APPENDIX I

In this paragraph we explain the code written to implement the Function Modules used to customize the BPS Exit Function:

- Function Module Z_P_BPSEXT_DATASLICE_ACT_INIT
- Function Module Z_P_BPSEXT_DATASLICE_ACT_EXEC

In addition, you should also implement some messages (with SE80). Here we used the following messages, that you can reproduce at your convenience:

Class	Number	Text
ZP_BPS_01	062	DataSlice &1 NOT found in Planning Area &2: Check Customizing
ZP_BPS_01	063	Invalid Parameter DS_STATE: check Area &1, Level &2, Func &3, Group &4
ZP_BPS_01	064	DataSlice &1 of Area &2 has been DE-Activated
ZP_BPS_01	065	DataSlice &1 of Area &2 has been Activated
ZP_BPS_01	066	DataSlice &1 of Area &2 is IN-Active: no changes will be done
ZP_BPS_01	067	DataSlice &1 of Area &2 is Active: no changes will be done

Function Module Z_P_BPSEXT_DATASLICE_ACT_INIT

This kind of FM is, generally, optional when implementing BPS functions of type EXIT. Here it’s used in order to activate or de-activate Dataslices, according to the parameters defined in Parameter Group.

Notes	ABAP Code

FM Importing and Exporting parameters declaration	
	```
	FUNCTION Z_P_BPSEXT_DATASLICE_ACT_INIT.
	*"----------------------------------------------------
	*"*"Local Interface:
	*"  IMPORTING
	*"     REFERENCE(I_AREA) TYPE  UPC_Y_AREA
	*"     REFERENCE(I_PLEVEL) TYPE  UPC_Y_PLEVEL
	*"     REFERENCE(I_PACKAGE) TYPE  UPC_Y_PACKAGE
	*"     REFERENCE(I_METHOD) TYPE  UPC_Y_METHOD
	*"     REFERENCE(I_PARAM) TYPE  UPC_Y_PARAM
	*"     REFERENCE(IT_EXITP) TYPE  UPF_YT_EXITP
	*"     REFERENCE(ITO_CHASEL) TYPE  UPC_YTO_CHASEL
	*"     REFERENCE(ITO_CHA) TYPE  UPC_YTO_CHA
	*"     REFERENCE(ITO_KYF) TYPE  UPC_YTO_KYF
	*"  EXPORTING
	*"     REFERENCE(ETO_CHAS) TYPE  SORTED TABLE
	*"     REFERENCE(ET_MESG) TYPE  UPC_YT_MESG
	*"----------------------------------------------------
	```

Notes	ABAP Code
Local Data (variables and internal tables) declarations	see code below

```
* wa to be used for APPEND Messages ...
  DATA: wa_et_mesg    TYPE upc_ys_mesg.
  DATA: fl_stop       TYPE C.
* Index on itab in LOOP
  DATA: my_tabix LIKE sy-tabix.
  DATA: wa_count TYPE i.

* Internal Tables and Header Lines
  DATA: tb_UPC_DATASLICET TYPE UPC_DATASLICET OCCURS 0.
  DATA: wa_UPC_DATASLICET LIKE LINE
        OF tb_UPC_DATASLICET.

  DATA: wa_UPC_DATASLICE TYPE UPC_DATASLICE.

* Parameters of Parameters Group
  DATA: wa_INACTIVE TYPE C.
  DATA: wa_AREA     TYPE UPC_Y_AREA.
  DATA: wa_PATH     TYPE UPC_Y_DATASLICETXT.
  DATA: ls_exitp    TYPE upf_ys_exitp.

  TYPES: BEGIN OF YS_SLICE,
           SORT TYPE UPC_DATASLICE-SORT,
           GUID TYPE UPC_DATASLICE-GUID,
           INACTIVE TYPE UPC_DATASLICE-INACTIVE,
           TEXT TYPE UPC_DATASLICET-TEXT,
           T_OPTIOS TYPE UPC_YT_OPTIOS,
           R_CHASEL TYPE REF TO IF_SEM_CHASEL,
           STATUS(1) TYPE C,
         END OF YS_SLICE.

  DATA: wa_SLICE TYPE YS_SLICE.

  TYPES: YTO_SLICE TYPE SORTED TABLE OF YS_SLICE
                   WITH UNIQUE KEY SORT.


  TYPES: BEGIN OF YS_AREA_SLICE,
           AREA TYPE UPC_Y_AREA,
           TO_SLICE TYPE YTO_SLICE,
           ENQMODE TYPE ENQMODE,
         END OF YS_AREA_SLICE.

  DATA: ES_AREA_SLICE TYPE YS_AREA_SLICE.
```

Notes | **ABAP Code**

Read DS_STATE Parameter defined in Parameter Group and check it's:

- **A** → Activation of the Dataslice
- **D** → De-Activation of the Dataslice

```
* Get State / Param of DataSlices
  READ TABLE it_exitp INTO ls_exitp
    WITH KEY parnm = 'DS_STATE'.

  IF sy-subrc = 0.
    CASE ls_exitp-chavl.
      WHEN 'A'.
*       DataSlices must be Activated.
        wa_INACTIVE = ''.
      WHEN 'D'.
*       DataSlices must be DE-Activated.
        wa_INACTIVE = 'X'.
      WHEN OTHERS.
        wa_et_mesg-MSGID = 'ZP_BPS_01'.
        wa_et_mesg-MSGTY = 'E'.
        wa_et_mesg-MSGNO = '063'.
        wa_et_mesg-MSGV1 = I_AREA.
        wa_et_mesg-MSGV2 = I_PLEVEL.
        wa_et_mesg-MSGV3 = I_METHOD.
        wa_et_mesg-MSGV4 = I_PARAM.
        append wa_et_mesg to et_mesg.
        fl_stop = 'X'. EXIT.
    ENDCASE.
  ELSE.
    wa_et_mesg-MSGID = 'ZP_BPS_01'.
    wa_et_mesg-MSGTY = 'E'.
    wa_et_mesg-MSGNO = '063'.
    wa_et_mesg-MSGV1 = I_AREA.
    wa_et_mesg-MSGV2 = I_PLEVEL.
    wa_et_mesg-MSGV3 = I_METHOD.
    wa_et_mesg-MSGV4 = I_PARAM.
    append wa_et_mesg to et_mesg.
    fl_stop = 'X'. EXIT.
  ENDIF.
```

Read DS_AREA and DS_PATH values from Parameter Group.

Since DS_AREA and DS_PATH can contain wildcards (*) replace with (%) in order to use LIKE operator in WHERE clause.

```
* Get Area of DataSlices
  READ TABLE it_exitp INTO ls_exitp
    WITH KEY parnm = 'DS_AREA'.
  IF sy-subrc = 0.
    wa_AREA = ls_exitp-chavl.
    OVERLAY wa_AREA WITH '%%%%%%%%' ONLY '*'.
  ENDIF.

* Get Path (on Text Fields) of DataSlices
  READ TABLE it_exitp INTO ls_exitp
    WITH KEY parnm = 'DS_PATH'.
  IF sy-subrc = 0.
    wa_PATH = ls_exitp-chavl.
    OVERLAY wa_PATH WITH '%%%%%%%%%%%%%%%%%%%%'
       ONLY '*'.
  ENDIF.
```

Get from DB Table UPC_DATASLICET all the Dataslice matching selection condition defined by DS_AREA and DS_PATH

```
  SELECT *
    FROM UPC_DATASLICET
    INTO TABLE tb_UPC_DATASLICET
   WHERE LANGU = sy-LANGU
     AND AREA  LIKE wa_AREA
     AND TEXT  LIKE wa_PATH
  ORDER BY AREA ASCENDING.
```

Notes | **ABAP Code**

For each Dataslice:

```
  LOOP AT tb_UPC_DATASLICET INTO wa_UPC_DATASLICET.
```

- Update Dataslice definition in BPS Buffer

```
    AT NEW AREA.
*     Get DataSlice State in Buffer ...
      PERFORM BUFFER_DATASLICE_GET
      IN PROGRAM SAPLUPC_DATASLICE
        USING wa_UPC_DATASLICET-AREA
              'E'
     CHANGING ES_AREA_SLICE.
    ENDAT.
```

- Get the Dataslice status from DB Table UPC_DATASLICE

```
    SELECT SINGLE *
      INTO wa_UPC_DATASLICE
      FROM UPC_DATASLICE
     WHERE AREA = WA_UPC_DATASLICET-AREA
       AND SORT = wa_UPC_DATASLICET-SORT.

    IF sy-dbcnt = 1 AND sy-subrc = 0.
```

- If the Status is different from the one defined in BPS Parameter Group, switch …

```
      IF wa_UPC_DATASLICE-INACTIVE <> wa_INACTIVE.
*       State has to be changed ...

        LOOP AT ES_AREA_SLICE-TO_SLICE INTO wa_SLICE
          WHERE SORT = wa_UPC_DATASLICE-SORT
            AND GUID = wa_UPC_DATASLICE-GUID.

          wa_SLICE-INACTIVE = wa_INACTIVE.
          wa_SLICE-STATUS   = 'M'.
          MODIFY ES_AREA_SLICE-TO_SLICE FROM wa_SLICE.
        ENDLOOP.
```

… and “refresh” also the BPS buffer definition of the Dataslice. This step is fundamental to have an on-line effect.

```
        PERFORM BUFFER_DATASLICE_UPDATE
        IN PROGRAM SAPLUPC_DATASLICE
       CHANGING ES_AREA_SLICE.

        CALL FUNCTION 'UPC_DATASLICE_COMMIT'.
        CALL FUNCTION 'UPC_DATASLICE_SAVE'.

        CLEAR ES_AREA_SLICE.

        wa_et_mesg-MSGID = 'ZP_BPS_01'.
        wa_et_mesg-MSGTY = 'I'.
        IF wa_INACTIVE = 'X'.
          wa_et_mesg-MSGNO = '064'.
        ELSE.
          wa_et_mesg-MSGNO = '065'.
        ENDIF.
        wa_et_mesg-MSGV1 = wa_UPC_DATASLICET-TEXT.
        wa_et_mesg-MSGV2 = wa_UPC_DATASLICET-AREA.
        append wa_et_mesg to et_mesg.
        wa_count = wa_COUNT + 1.

      ELSE.
```

- If the Status matched the one in Parameter Group no changes have to be done. Just provide a message about.

```
*       State is identical ... no changes
        wa_et_mesg-MSGID = 'ZP_BPS_01'.
        wa_et_mesg-MSGTY = 'I'.
        IF wa_INACTIVE = 'X'.
          wa_et_mesg-MSGNO = '066'.
        ELSE.
          wa_et_mesg-MSGNO = '067'.
        ENDIF.
        wa_et_mesg-MSGV1 = wa_UPC_DATASLICET-TEXT.
        wa_et_mesg-MSGV2 = wa_UPC_DATASLICET-AREA.
        append wa_et_mesg to et_mesg.
      ENDIF.
    ELSE.
```

Notes	ABAP Code
	```
	*      Error DATASLICE NOT FOUND
	       wa_et_mesg-MSGID = 'ZP_BPS_01'.
	       wa_et_mesg-MSGTY = 'E'.
	       wa_et_mesg-MSGNO = '062'.
	       wa_et_mesg-MSGV1 = wa_UPC_DATASLICET-TEXT.
	       wa_et_mesg-MSGV2 = wa_UPC_DATASLICET-SORT.
	       wa_et_mesg-MSGV3 = wa_AREA.
	       append wa_et_mesg to et_mesg.
	       EXIT.
	     ENDIF.
	   ENDLOOP.

	ENDFUNCTION.
	```

Function Module Z_P_BPSEXT_DATASLICE_ACT_EXEC

Notes	ABAP Code

This FM, generally, provides data creation / changes.
 	
Since the fact that it is mandatory to customize Parameter Group related to BPS EXIT functions is empty: XTH_DATA remains unchanged.
	```
	FUNCTION Z_P_BPSEXT_DATASLICE_ACT_EXEC.
	*"----------------------------------------------------
	*"*"Local Interface:
	*"  IMPORTING
	*"     REFERENCE(I_AREA) TYPE  UPC_Y_AREA
	*"     REFERENCE(I_PLEVEL) TYPE  UPC_Y_PLEVEL
	*"     REFERENCE(I_PACKAGE) TYPE  UPC_Y_PACKAGE
	*"     REFERENCE(I_METHOD) TYPE  UPC_Y_METHOD
	*"     REFERENCE(I_PARAM) TYPE  UPC_Y_PARAM
	*"     REFERENCE(IT_EXITP) TYPE  UPF_YT_EXITP
	*"     REFERENCE(ITO_CHASEL) TYPE  UPC_YTO_CHASEL
	*"     REFERENCE(ITO_CHA) TYPE  UPC_YTO_CHA
	*"     REFERENCE(ITO_KYF) TYPE  UPC_YTO_KYF
	*"  EXPORTING
	*"     REFERENCE(ET_MESG) TYPE  UPC_YT_MESG
	*"  CHANGING
	*"     REFERENCE(XTH_DATA) TYPE  HASHED TABLE
	*"----------------------------------------------------

	ENDFUNCTION.
	```

Related Content

SAP Help on Dataslice
SAP Help on Buffer Concept (BW-BPS)
SAP Help on EXIT Function

Disclaimer and Liability Notice

Copyright