ISPF Programmer's Guide

PREV
10 Panels – create and use

NEXT
12 Tables – Create and edit

11 Skeletons – Design and use

In this chapter, you will learn how you can create JCL members and other data elegantly by using ISPF skeletons. ISPF skeletons are members in a PDS/PDSE that are created with the editor. They are masks which are used for the management of any data structures. Skeleton members consist of basic texts, skeleton control statements and variable names. The variable names are replaced during processing of skeletons by the **ISPF file-tailoring services** by the actual values which are set in the processing REXX procedure. As with the panels, the variable names must not be longer than eight characters.

11.1 CREATING SKELETONS

For the creation of skeletons, the following commands are available:

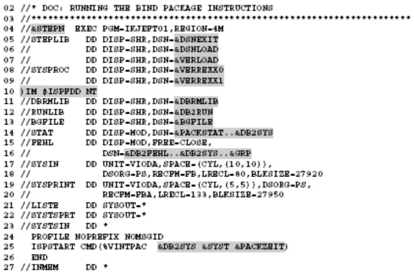
Table 11.1: Commands to create skeletons

Command	Function
)BLANK	)BLANK 5 inserts 5 blank lines into the skeleton, which is being edited.
)CM	With)CM, comment lines can be defined in the skeleton. These lines are not copied to the output component.
)DEFAULT	To control the skeleton processing, seven special characters are reserved by default. These are the characters: **), &, ?, !, <, |, und >**. With the)DEFAULT command these characters can be changed temporarily
)DOT	Using these two statements, a loop can be formed.
)ENDDOT	
)IM	This statement inserts another skeleton. The variables of the inserted skeleton will be dissolved and the result is added to the output component.
)SEL	With)SEL a condition can be queried. If this condition is satisfactory, all the rows are
)ENDSEL	processed until the next)ENDSEL.
)SET	Assignments can be made by using this instruction.
)TB,)TBA	Tab stops can be set by using this instruction.

Example:

The following skeleton adds a step into a job which executes instructions for DB2 BIND PACKAGE commands. The BIND PACKAGE instructions will not be executed in this job step; they are executed in the called REXX procedure VINTPAC.

JCL 11.6: Definition of a skeleton



```
02 //* DOC: RUNNING THE BIND PACKAGE INSTRUCTIONS
03 //*********************************************************************
04 //&STEPN  EXEC PGM=IKJEFT01,REGION=4M
05 //STEPLIB   DD DISP=SHR,DSN=&DSNEXIT
06 //          DD DISP=SHR,DSN=&DSNLOAD
07 //          DD DISP=SHR,DSN=&VERLOAD
08 //SYSPROC   DD DISP=SHR,DSN=&VERREXX0
09 //          DD DISP=SHR,DSN=&VERREXX1
10 )IM $ISPFDD NT
11 //DBRMLIB   DD DISP=SHR,DSN=&DBRMLIB
12 //RUNLIB    DD DISP=SHR,DSN=&DB2RUN
13 //BGFILE    DD DISP=SHR,DSN=&BGFILE
14 //STAT      DD DISP=MOD,DSN=&PACKSTAT..&DB2SYS
15 //FEHL      DD DISP=MOD,FREE=CLOSE,
16 //             DSN=&DB2FEHL..&DB2SYS..&GRP
17 //SYSIN     DD UNIT=VIODA,SPACE=(CYL,(10,10)),
18 //             DSORG=PS,RECFM=FB,LRECL=80,BLKSIZE=27920
19 //SYSPRINT  DD UNIT=VIODA,SPACE=(CYL,(5,5)),DSORG=PS,
20 //             RECFM=FBA,LRECL=133,BLKSIZE=27950
21 //LISTE     DD SYSOUT=*
22 //SYSTSPRT  DD SYSOUT=*
23 //SYSTSIN   DD *
24   PROFILE NOPREFIX NOMSGID
25   ISPSTART CMD(%VINTPAC  &DB2SYS &SYST &PACKZEIT)
26   END
27 //INMEM     DD *
```

Remarks:

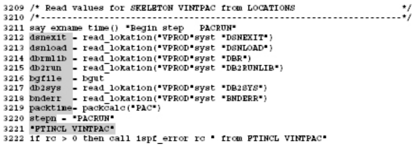
– I have grayed all relevant places for a skeleton definition.

– In line 10 the skeleton $ISPFDD is inserted. This skeleton contains the necessary DD statements for execution of REXX procedures in an ISPF batch environment.

– If a skeleton variable should be connected to another skeleton variable, a point is used as concatenation operator. If the resulting string is a DSN, an additionally point will be inserted for separating the single DSN qualifiers. DSNs are built in the lines 14 and 16. Therefore, here two consecutive dots are used to assemble DSNs.

Here are the statements of the REXX procedure VINTSN that processes this skeleton:

Coding 11.1: Skeleton processing



```
3209 /* Read values for SKELETON VINTPAC from LOCATIONS                    */
3210 /*--------------------------------------------------------------------*/
3211 say exname time() "Begin step   PACRUN"
3212 dsnexit  = read_lokation("VPROD"syst "DSNEXIT")
3213 dsnload  = read_lokation("VPROD"syst "DSNLOAD")
3214 dbrmlib  = read_lokation("VPROD"syst "DBR")
3215 db2run   = read_lokation("VPROD"syst "DB2RUNLIB")
3216 bgfile   = bgut
3217 db2sys   = read_lokation("VPROD"syst "DB2SYS")
3218 bnderr   = read_lokation("VPROD"syst "BNDERR")
3219 packtime= packcalc("PAC")
3220 stepn  = "PACRUN"
3221 "FTINCL VINTPAC"
3222 if rc > 0 then call ispf_error rc " from FTINCL VINTPAC"
```

In this part of the program, most skeleton variables are filled with values. Some have already been set earlier in the procedure. When the statement FTINCL VINTPAC in line 3221 is executed, the following happens:

– The member VINTPAC will be searched in the data set chain associated with the DD name ISPSLIB and inserted in an internal area.

– Thereafter, the file-tailoring service is called. This processes the skeleton line by line.

– The contents of the variables from the REXX/ISPF variable pool replaces the variables.

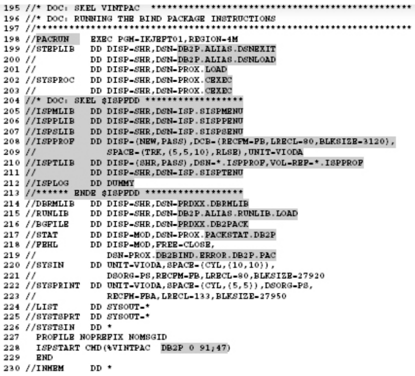
– Any existing file-tailoring statements are executed. In this skeleton only one)IM statement is contained. The member $ISPFDD is searched in the ISPSLIB chain and inserted in the JCL stream. The variables contained in the member will be replaced by the actual values.

– When the file-tailoring service has conducted all processing, the result is added to the internal data stream that has been opened with the FTOPEN statement.

– When all file-tailoring work that is necessary to form a JCL member is finished in the procedure VINTSN, the created JCL member is saved using the command FTCLOSE NAME(name) LIBRARY(ddname). If ddname is not specified **ISPFILE** is used.

The job step created by the file-tailoring service looks as follows:

JCL 11.7: Job step created by the file-tailoring service



```
//* DOC: SKEL VINTPAC  ***********************************************
//* DOC: RUNNING THE BIND PACKAGE INSTRUCTIONS
//*******************************************************************
//PACRUN     EXEC PGM=IKJEFT01,REGION=4M
//STEPLIB    DD DISP=SHR,DSN=DB2P.ALIAS.DSNEXIT
//           DD DISP=SHR,DSN=DB2P.ALIAS.DSNLOAD
//           DD DISP=SHR,DSN=PROX.LOAD
//SYSPROC    DD DISP=SHR,DSN=PROX.CEXEC
//           DD DISP=SHR,DSN=PROX.CEXEC
//* DOC: SKEL $ISPFDD ******************
//ISPMLIB    DD DISP=SHR,DSN=ISP.SISPMENU
//ISPPLIB    DD DISP=SHR,DSN=ISP.SISPPENU
//ISPSLIB    DD DISP=SHR,DSN=ISP.SISPSENU
//ISPPROF    DD DISP=(NEW,PASS),DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120),
//              SPACE=(TRK,(5,5,10),RLSE),UNIT=VIODA
//ISPTLIB    DD DISP=(SHR,PASS),DSN=*.ISPPROF,VOL=REF=*.ISPPROF
//           DD DISP=SHR,DSN=ISP.SISPTENU
//ISPLOG     DD DUMMY
//****** ENDE $ISPFDD ******************
//DBRMLIB    DD DISP=SHR,DSN=PRDXX.DBRMLIB
//RUNLIB     DD DISP=SHR,DSN=DB2P.ALIAS.RUNLIB.LOAD
//BGFILE     DD DISP=SHR,DSN=PRDXX.DB2PACK
//STAT       DD DISP=MOD,DSN=PROX.PACKSTAT.DB2P
//FEHL       DD DISP=MOD,FREE=CLOSE,
//              DSN=PROX.DB2BIND.ERROR.DB2P.PAC
//SYSIN      DD UNIT=VIODA,SPACE=(CYL,(10,10)),
//              DSORG=PS,RECFM=FB,LRECL=80,BLKSIZE=27920
//SYSPRINT   DD UNIT=VIODA,SPACE=(CYL,(5,5)),DSORG=PS,
//              RECFM=FBA,LRECL=133,BLKSIZE=27950
//LIST       DD SYSOUT=*
//SYSTSPRT   DD SYSOUT=*
//SYSTSIN    DD *
  PROFILE NOPREFIX NOMSGID
  ISPSTART CMD(%VINTPAC  DB2P 0 91;47)
  END
//INMEM      DD *
```

In the previous image, I grayed all the locations where values were inserted by variable substitution and all inserted rows.

11.2 STEPS TO USE THE FILE-TAILORING SERVICE

1. Use the ISPF editor to create a member in a PDS that is assigned under the DD name ISPSLIB for processing skeletons by a REXX procedure.

2. Create a REXX procedure where the variables are set used by the file-tailoring service to replace the corresponding variables of the skeletons.

3. Use in the REXX program the FTINCL service to insert a skeleton.

4. Use the FTCLOSE NAME**(name)** service to save the member **name.**

Tip:
The file-tailoring service can also be used to generate any arbitrary data set.

Recommended / Playlists / History / Topics / Settings / Get the App / Sign Out

PREV
10 Panels – create and use

NEXT
12 Tables – Create and edit