

12 Tables - Create and edit

Here you will learn how to create, edit and view ISPF tables. The table service of ISPF provides the ability to create tables, to edit and to use them for your daily work. A row in the ISPF table consists of columns that are named by ISPF variables. Each line can be addressed using ISPF table service commands. After carrying out such a command, the names of the variables in the manufacturing procedure are known.

12.1 LOCATIONS FOR TABLES

Tables can be created either temporarily or permanently. A temporary table only exists in virtual memory. It cannot be saved in a permanent data set.

For processing, the entire table always resides in **virtual** memory. This means that an ISPF table is limited in its size by the available virtual memory. Normally tables are stored in memory above the 16-megayte-line. This allows ISPF tables to become quite large.

12.2 READING ISPF TABLES

A table is read from the ISPTLIB data set chain with the TBOPEN statement into the memory or it is created there temporarily. When a table is read from ISPTLIB with the WRITE option, then an ENQ is set on the member in which this table resides. This prevents that this table can be read a second time. The ENQ is cancelled when the table is closed with TBCLOSE. When reading with the NOWRITE option, the ENQ is only active during the read operation. Under the DD name ISPTLIB several PDS/PDSE are normally allocated. When executing the TBOPEN, the member is searched in this data set chain. The first occurring member of the name will be loaded.

12.3 WRITING ISPF TABLES

When writing ISPF table members, you cannot use the DD name ISPTLIB because under this DD name usually a chain of PDS/PDSE is allocated. Moreover, we have learned above, that we cannot write to a DD, which addresses a concatenated data set. The DD name ISPTABL is used to write tables. Therefore, always only one data set is allocated to ISPTABL. If this applies, you must consider that the following question naturally arises: How can I maintain a table if I cannot write back to the file of which I have read?

A small **trick** solve this problem:

- You can read all the tables that are included in the PDS/PDSE concatenation of about the DD name ISPTLIB.
- Tables can only be written into a data set that is associated with the DD name ISPTABL.
- When the data set that is allocated with the DD name ISPTABL to write tables is also allocated as first data set in the ISPTLIB chain, then a recently written table will always be loaded with its latest contents.

This allocation looks in practice in my TSO user:

| T1T105 | SHR, KEEP | > | ISPTABL | LANZT.ISPF.TLIBTAB.LPRT |
|--------|-----------|---|---------|----------------------------|
| | | > | ISPTLIB | |
| T1T105 | SHR, KEEP | > | | LANZT. ISPF. TLIBTAB. LPRT |
| PLPRT1 | SHR, KEEP | > | | SYST1.TSO.ISPTLIB |
| PLPRT1 | SHR, KEEP | > | | SYST1.JCLPLUS.ISPTLIB |
| OST0A9 | SHR, KEEP | > | | SYS1.DGTTLIB |
| T1S001 | SHR, KEEP | > | | SYSPT.CAI.CAIISPT |
| OST0A9 | SHR, KEEP | > | | ISP.SISPTENU |
| | | | | |

i Rule:

When I write a table (whether new or read before), then it will be written via ISPTABL in the LANZT.ISPF.TILBTAB.LPRT data set. When I read them afterwards, it is first found in the same physical data set under the DD name ISPTLIB and read from there. In order for the ongoing maintenance of a table will guaranteed.

12.4 COMMANDS OF THE TABLE SERVICES

In ISPF, a number of commands for working with tables is available. We distinguish the commands that relate to an entire table and those, which relate to the processing within the tables:

Table 12.1: Table services commands that refer to the entire table

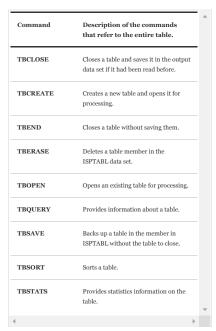
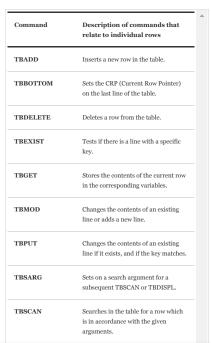


Table 12.2: Table service commands that relate to individual rows





Again, I will refrain from describing the above commands in detail. Instead, the following examples will show you how to create a table and how to use it.

12.5 EXAMPLE OF WORKING WITH TABLES

As I mentioned on several occasions, I wrote an ISPF application which displays the DSNs of the most recently edited data sets in a panel. The main program of this application is SLE which stands for Smart Last Edit. It is contained in the collection of the SMART ISPF utilities. All programs of this application are written in REXX. The following table shows the most important components that belong to this application:

Table 12.3: Components of the SLE application

| Name | Туре | Function | |
|----------|-----------------|--|--|
| SLE | Main program | This program is called as ISPF function. It displays the panel SLEP1 and performs the work initiated there. | |
| #IMACROA | Edit macro | If an edit session is started, this macro checks whether the IMACRO option is already set for the edited data set in the edit profile. If this is not the case, the option IMACRO is set to #IMACRO1. | |
| #IMACRO1 | Edit macro | This macro defines an ALIAS so, that at the end of the edit session #IMACRO2 is called. | |
| #IMACRO2 | Edit macro | This macro will be called automatically when an edit session ends. It inserts the DSN of the currently edited data set into the ISPF table \$SLETAB. At the next call of SLE, the name of the recently edited data set is contained in the table \$SLETAB and it will be displayed in the panel SLEP1. | |
| SLEP1 | ISPF panel | This is the main panel for execution of the SLE application. It shows the table \$\$LETAB and offers the ability to perform some actions with the shown data sets. | |
| | | | |

Many help panels are defined to assist the user in the working with the SLE application. However, these panels are not important for the function of the application and are therefore not named here.

Here the part of the edit macro #IMACRO2, which shows the updating (or creation) of the ISPF table \$SLETAB:

Program 12.1: Excerpt from edit macro #IMACRO2

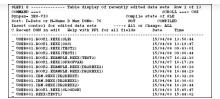
```
1 1 educar c; then dum - sident 'shear', c dee dem - sident 'shear', c dee dem - sident 's sequential c very community of the dem - sident 's sequential c very (server) dumber of the sident 's sequential c very (server), sident c very (server), s
           /* Insert the name of the edited data set into table $SLETAR *
           Select PRINC - 0 then do
"Shen Princ" - 0 then do
"TREDD" 'tabname' DEDER / "Updata table SELETAE "/
"TREDD 'tabname' EREAGE (DATE C.D.THE C.D."
"TREDT" 'tabname' LIERARY (ISPTAEL) REPLOOF PAD (100)"
            ond
when openic - 8 then do /* Table SILETAB does not exist. Create it */
"TROCEATE 'tabname EFFS(16S), NAMES(PUNCTION, DATE, TIME) *
TIBADD 'tabname OEDE
"TROCEOE" 'tabname 'LIBRARY(ISPTABL) REPLOOP PAD(100) *
           end
when openrc > s then do
zedlmsg = exname*:* zerrim
*serwsg Msg(ISR2001)*
           Lines
                                                   Explanations
                                                   Table $SLETAB is opened. The RC is saved in
                                                  Some table variables are set.
           68
                                                  When the table $SLETAB exists, it will be
           70-74
                                                  updated and written back to ISPTABL.
                                                   When the table $SLETAB does not exist it will
          75-79
                                                   be created and written into ISPTABLE as new
                                                   member which then contains one row of data.
   Here an excerpt of the panel description of panel SLEP1 where the )MODEL line
   defines the structure of the displayed rows of the table $SLETAB.
   Coding 12.1:Excerpt from panel definition of the panel SLEP1
 21 "CRecent DGN in edit WHelp with FFI for all fields 22 ft-0 |
22 ft-0 |
23 MODEL |
24 cf.dam |
25 JINT |
26 JEVER - '(dynpan lesort maxiows alleha function)' |
27 Jelp - GERTI |
28 JOSTFOW - 6line |
29 VOST (EGROT, alleha) PROFILE
                                                                                                                                                ;date ;time
           Lines
                                                   Explanation
                                                   Here the model definition segment begins.
                                                   This line can be followed by up to 8
                                                  definitions of data structures.
                                                   Here only one line structure is defined. It
           24
                                                  contains the names and positions where the
                                                   contents of the variables of the table
                                                   $SLETAB are displayed. Because in the
                                                   previous panel definitions already four Z-
                                                   variables are set, the Z-variable in line 24 for
                                                   the name FUNCTION is assigned by the
                                                    .ZVARS definition.
   The complete definition of panel SLEP1 is found in Program 10.1: Panel
  definition of panel SLEP1 on page 194.
   The TBDISPL command to display the panel SLEP1 together with the table
  $SLETAB is shown in the following excerpt from program SLE:
  Coding 12.2: Excerpt from program SLE showing the TBDISPL command of
```

Find answers on the fly, or master something new. Subscribe today. See pricing options.

Here now a display of the panel SLEP1 containing the data of the table

Screen 12.1: Display of panel SLEP1 produced by the program SLE

\$SLETAB:



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



NEXT 13 Variables – Definition and using