PREV
12 Tables – Create and edit

NEXT
14 Edit macros – Create and apply

13 Variables – Definition and using

In this chapter, you will learn how to define and use ISPF variables. ISPF variables are either automatically known in REXX programs or they can be made known there.

Advice:

If you intend to use a defined variable in REXX source code and in the called ISPF services, then the variable name must not be longer than **eight** characters.

The variables of the ISPF services are an important part of ISPF. Dialog variables are the primary communicator between the individual components of ISPF. Programs, procedures, panels, tables and skeletons can all share the variables from the three ISPF variable pools. The three ISPF variable pools are:

– Function pool for implicit variables.

– Shared pool.

– Profile pool.

The variables are combined in groups, which are associated with a particular service. Many variables are set only by certain ISPF services. Other variables must be filled with appropriate values before a service that requires these variables is called. A ISPF application ID is assigned to each pool of variables.

13.1 VARIABLES POOLS

As mentioned above, the ISPF variables are arranged in variables pools with the following subdivisions:

13.1.1 Function pool

These variables are only known in the currently running function or group of procedures. These include, for example, all variables in a REXX procedure whch names are not longer than eight characters, and the variables of panels, skeletons etc. These variables are called **function variables**.

Each function has its own set of variables. If an item within the ISPF in different logical screens is launched parallel, then the variable exist with the same name several times, but with different contents. These variables can only be exchanged with other functions when you write them in a function in the shared pool and are read in the other function from there.

When a new function will be started, ISPF creates a variables pool for this function. This pool of variables is deleted at the end of the function. A function is a program or a procedure that can be called with the SELECT service. This means you can exchange variables between these programs or procedures only via the shared or the profile pool.

13.1.2 Shared pool

The shared pool contains variables that are used within an application. A variable in this pool is called **shared variable**. The SELECT service creates a shared pool when executing the commands ISPSTART or ISPF. The same happens when a SELECT is performed using the options NEWAPPL or NEWPOOL .

One function may use the VPUT service to copy shared variables into the shared pool, and another function in the same application can use the VGET service to

The variables in this pool will be saved from session to session for each application. This means that the variables of this application are written in a member which is associated with the application in the profiles data set. When the application is started later again, the corresponding member will be read.

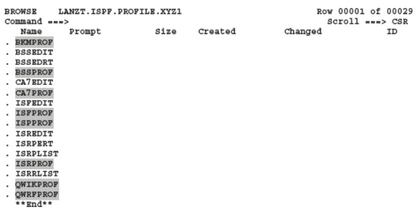
If you need these variables in a function, you have to read them using the VGET service. To save them so that they are available again the next time you start a function, they must be re-written using the VPUT service into the profile pool before leaving the function.

Thus, the variables of profiles pools can be made available from session to session and from application to application. Of course, a data set must exist in which these variables are saved. This data set must be assigned under the DD name ISPPROF as PDS/PDSE before starting the ISPF. **If this data set is not assigned, ISPF cannot be started**. As mentioned above, the variables in the profiles pool are stored **separately for each ISPF application** into a member in the profile data set. The name of this member is formed as follows:

xxxx **PROF**, where xxxx is the up to four digits long name of the application. The profile members are created and processed as ISPF tables.

The following figure shows an excerpt from the list of members in the ISPPROF data set of my TSO user:

Screen 13.1: Members of an ISPPROF data set



```
BROWSE    LANZT.ISPF.PROFILE.XYZ1                         Row 00001 of 00029
Command ===>                                                  Scroll ===> CSR
   Name     Prompt          Size    Created         Changed           ID
. BKMPROF
. BSSEDIT
. BSSEDRT
. BSSPROF
. CA7EDIT
. CA7PROF
. ISFEDIT
. ISFPROF
. ISPPROF
. ISREDIT
. ISRPERT
. ISRPLIST
. ISRPROF
. ISRRLIST
. QWIKPROF
. QWRFPROF
  **End**
```

The application dependent members are grayed.

13.2 SAVING THE PROFILE MEMBERS

If an ISPF application is terminated normally, their profile variables are written back to the appropriate member of the profile data set. The member ISPPROF will only be written back if the ISPF itself terminates.

Attention:

If ISPF, from whatever reason, is not properly completed or is aborted, then the currently active profile members will **not** be written back. This means that after a restart of ISPF, all changes made in the profile variables before the demolition are lost. The same applies if a single application aborts. In this case, the profile member containing the application variables will not be secured.

13.3 CREATING PROFILE MEMBERS

When ISPF or a new application starts, then ISPF searches for a profile member as

follows:

1First, the user profile associated with the DD name ISPPROF is searched for the member xxxxPROF. If the member is found there, then it is loaded and the profile variables contained therein are available for the application. This means you can now read variables with VGET and write back with VPUT.

2If the profile member is not found in the ISPPROF chain, then the member will be searched in the ISPTLIB chain. If it is found there, it will used and written back in the ISPPROF data set when the application or ISPF is closed.

3.When also on the ISPTLIB chain no fitting member for the launched application is found, the default profile member will be read from ISPPROF and copied to the variable pool for this application. When closing the application, its profile member is written into the ISPPROF data set.

Consideration:

ISPF will be started at the end of the LOGON procedure. This can be done with three different commands:

1ISPSTART PANEL(name) NEWAPPL(applid).

2PDF

3ISPF

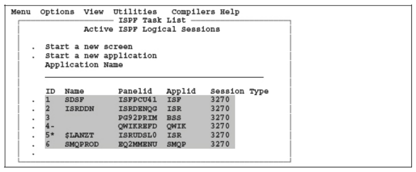
This way of calling the ISPF causes the standard APPLID in the ISPF to be **ISR**. This means that the member **ISRPROF** is used as standard ISPF profile member. Therefore, if you want to operate your ISPF with the APPLID ISP, then you would have to start your ISPF with the following command:

ISPSTART PANEL(ISP@PRIM) NEWAPPL(ISP).

13.4 DISPLAY OF ACTUALLY USED PROFILE MEMBER

I have launched several ISPF applications in my TSO user. Then I looked which profile members are loaded now. Here is the first indication of the started applications. I created this display using the ISPF command **SWAP LIST**.

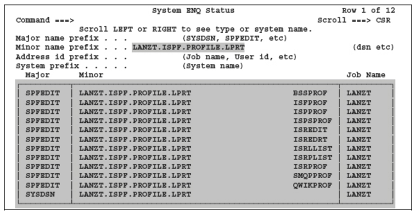
Screen 13.2: List of started ISPF applications



```
Menu  Options  View  Utilities   Compilers Help
 ┌──────────────── ISPF Task List ─────────────┐
 │           Active ISPF Logical Sessions       │
 │                                              │
 │  .  Start a new screen                       │
 │  .  Start a new application                  │
 │     Application Name                         │
 │     ____________________________________     │
 │                                              │
 │     ID  Name       Panelid   Applid   Session Type
 │  .  1   SDSF       ISFPCU41  ISF      3270    │
 │  .  2   ISRDDN     ISRDENQG  ISR      3270    │
 │  .  3              PG92PRIM  BSS      3270    │
 │  .  4-             QWIKREFD  QWIK     3270    │
 │  .  5*  $LANZT     ISRUDSL0  ISR      3270    │
 │  .  6   SMQPROD    EQ2MMENU  SMQP     3270    │
 │  .                                           │
 └──────────────────────────────────────────────┘
```

When these applications are launched, then all members belonging to these applications profiles must be loaded. This can be checked very easily by looking up which ENQs exist on the data set containing the profile members. The easiest way to check this is using the utility DDLIST. Therefore, I called DDLIST and entered ENQ in the command line. In the appearing panel, I entered as **minor name prefix** the DSN of my profile data set LANZT.ISPF.PROFILE.LPRT and pressed ENTER.

The following ENQs are displayed:



```
                         System ENQ Status                          Row 1 of 12
 Command ===>                                                    Scroll ===> CSR
           Scroll LEFT or RIGHT to see type or system name.
 Major name prefix . . .           (SYSDSN, SPFEDIT, etc)
 Minor name prefix . . . LANZT.ISPF.PROFILE.LPRT                      (dsn etc)
 Address id prefix . . .           (Job name, User id, etc)
 System prefix . . . . .           (System name)
   Major      Minor                                                Job Name
 ┌──────────┬───────────────────────────────────────────────┬─────────┐
 │ SPFEDIT  │ LANZT.ISPF.PROFILE.LPRT              BSSPROF  │ LANZT   │
 │ SPFEDIT  │ LANZT.ISPF.PROFILE.LPRT              ISFPROF  │ LANZT   │
 │ SPFEDIT  │ LANZT.ISPF.PROFILE.LPRT              ISPPROF  │ LANZT   │
 │ SPFEDIT  │ LANZT.ISPF.PROFILE.LPRT              ISPSPROF │ LANZT   │
 │ SPFEDIT  │ LANZT.ISPF.PROFILE.LPRT              ISREDIT  │ LANZT   │
 │ SPFEDIT  │ LANZT.ISPF.PROFILE.LPRT              ISREDRT  │ LANZT   │
 │ SPFEDIT  │ LANZT.ISPF.PROFILE.LPRT              ISRLLIST │ LANZT   │
 │ SPFEDIT  │ LANZT.ISPF.PROFILE.LPRT              ISRPLIST │ LANZT   │
 │ SPFEDIT  │ LANZT.ISPF.PROFILE.LPRT              ISRPROF  │ LANZT   │
 │ SPFEDIT  │ LANZT.ISPF.PROFILE.LPRT              SMQPPROF │ LANZT   │
 │ SPFEDIT  │ LANZT.ISPF.PROFILE.LPRT              QWIKPROF │ LANZT   │
 │ SYSDSN   │ LANZT.ISPF.PROFILE.LPRT                       │ LANZT   │
 └──────────┴───────────────────────────────────────────────┴─────────┘
```

As you can see, the corresponding profile members are loaded and ENQed for all open applications.

13.5 THE SYSTEM PROFILE POOL

ISPF contains own variables that can be used by the user, but not overwritten. These variables are always present. They are stored in the member ISPSPROF. This member is always loaded when ISPF starts. The system profiles pool is located behind the ISPF standard pool. Therefore, if you read a variable from the system profile pool, you can put it back in the standard profiles pool. However, you must observe the following:

>Attention:

When you read a variable from the system profile pool, you can change it and write it back. In this case, the changed variable be stored in the standard profile pool. Afterwards is it impossible to read the origin variable again from the system profile pool. If you delete the variable using the VERASE command, you can read then the origin variable from the system profile pool again.

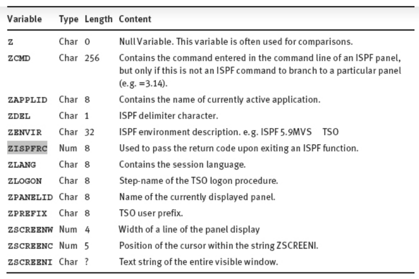
Tip:

In the brochure, **ISPF Dialog Developer's Guide and Reference** there are several attachments, where all standard ISPF variables are described.

13.5.1 Frequently used ISPF variables

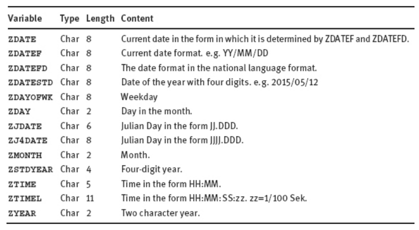
As mentioned above, many ISPF variables are only provided when certain services are running. These variables can be found in the description of the respective services. However, there is a whole range of ISPF variables that you can make good use of when creating applications in ISPF. If you want to use these variables in a REXX program, then you can read the appropriate variables using the command **ISPEXEC VGET (namel, name2, …)**. In the two tables below, I listed several ISPF system variables that are used most frequently:

13.5.1.1 General ISPF variables



Variable	Type	Length	Content
Z	Char	0	Null Variable. This variable is often used for comparisons.
ZCMD	Char	256	Contains the command entered in the command line of an ISPF panel, but only if this is not an ISPF command to branch to a particular panel (e.g. =3.14).
ZAPPLID	Char	8	Contains the name of currently active application.
ZDEL	Char	1	ISPF delimiter character.
ZENVIR	Char	32	ISPF environment description. e.g. ISPF 5.9MVS TSO
ZISPFRC	Num	8	Used to pass the return code upon exiting an ISPF function.
ZLANG	Char	8	Contains the session language.
ZLOGON	Char	8	Step-name of the TSO logon procedure.
ZPANELID	Char	8	Name of the currently displayed panel.
ZPREFIX	Char	8	TSO user prefix.
ZSCREENW	Num	4	Width of a line of the panel display
ZSCREENC	Num	5	Position of the cursor within the string ZSCREENI.
ZSCREENI	Char	?	Text string of the entire visible window.

13.5.1.2 System variables containing date and time in different variations



Variable	Type	Length	Content
ZDATE	Char	8	Current date in the form in which it is determined by ZDATEF and ZDATEFD.
ZDATEF	Char	8	Current date format. e.g. YY/MM/DD
ZDATEFD	Char	8	The date format in the national language format.
ZDATESTD	Char	8	Date of the year with four digits. e.g. 2015/05/12
ZDAYOFWK	Char	8	Weekday
ZDAY	Char	2	Day in the month.
ZJDATE	Char	6	Julian Day in the form JJ.DDD.
ZJ4DATE	Char	8	Julian Day in the form JJJJ.DDD.
ZMONTH	Char	2	Month.
ZSTDYEAR	Char	4	Four-digit year.
ZTIME	Char	5	Time in the form HH:MM.
ZTIMEL	Char	11	Time in the form HH:MM:SS:zz. zz=1/100 Sek.
ZYEAR	Char	2	Two character year.

13.5.2 Process ISPF variables

There are exactly three commands that you can use with REXX to work with ISPF variables. These are:

VGET	→	Read variables
VPUT	→	Write variables
VERASE	→	Delete variables

Formats of the three commands

VGET (name-list) ASIS | SHARED | PROFILE

ASIS	The variable will initially be searched in the shared pool. If it is not found there, the search is continued in the profile pool.
SHARED	The variable is searched only in the shared pool. If it is not found there, the command ends with a **RC=8**.
PROFILE	The variable will be searched only in profile pool. If it is not found there, the command ends with a **RC=8** and a variable of the same name in the shared pool will be deleted.

VPUT (name-list) ASIS | SHARED | PROFILE

ASIS	The variable is written to the pool, where it already exists. If it exists nowhere, it is written into the shared pool. If it already exists in the shared pool and in the profile pool, it is only changed in the shared pool.
SHARED	The variable is only written to the shared pool.
PROFILE	The variable is only written to the profile pool. A variable of the same name in the shared pool is deleted.

VERASE (name-list) ASIS | SHARED | PROFILE | BOTH

ASIS	The variable is deleted from the shared pool. If it

SHARED	The variable is deleted from the shared pool.
PROFILE	The variable is deleted from the profile pool.
BOTH	The variable is deleted from the shared and from the profile pool.

Remark:

All REXX variables whose name is not longer than eight characters are automatically known in **function pool**. If you want to write a REXX variable into the shared or profile pool, you must explicitly use a VPUT command.

Example:

I wrote a small REXX procedure to read and display some ISPF Z-variables:

Program 13.1: ZVARS – Display some ISPF Z-variables

```
/* DOC: ZVARS    REXX MAIN                                          */
/* DOC: Display some Z variables of ISPF                            */
/* © FRANZ LANZ 2015                                                */
/******************************************************************/
address "ISPEXEC"
"VGET (ZDATE ZDATE ZDATEF ZDATEFD ZDATESTD ZDAYOFWK ZDAY ZJDATE",
      "ZJ4DATE ZMONTH ZSTDYEAR ZTIME ZTIMEL ZYEAR ZUSER ZLOGON ",
      "ZSYSID ZAPPLID ZDEL ZENVIR ZLANG ZPANELID ZPREFIX)"

say "ZDATE    = "ZDATE
say "ZDATEF   = "ZDATEF
say "ZDATEFD  = "ZDATEFD
say "ZDATESTD = "ZDATESTD
say "ZDAYOFWK = "ZDAYOFWK
say "ZDAY     = "ZDAY
say "ZJDATE   = "ZJDATE
say "ZJ4DATE  = "ZJ4DATE
say "ZMONTH   = "ZMONTH
say "ZSTDYEAR = "ZSTDYEAR
say "ZTIME    = "ZTIME
say "ZTIMEL   = "ZTIMEL
say "ZYEAR    = "ZYEAR
say "ZUSER    = "ZUSER
say "ZLOGON   = "ZLOGON
say "ZSYSID   = "ZSYSID
say "ZAPPLID  = "ZAPPLID
say "ZDEL     = "ZDEL
say "ZENVIR   = "ZENVIR
say "ZLANG    = "ZLANG
say "ZPANELID = "ZPANELID
say "ZPREFIX  = "ZPREFIX
```

I executed this procedure and got the following result:

```
ZDATE    = 15/05/12
ZDATEF   = YY/MM/DD
ZDATEFD  = YY/MM/DD
ZDATESTD = 2015/05/12
ZDAYOFWK = Tuesday
ZDAY     = 12
ZJDATE   = 15.132
ZJ4DATE  = 2015.132
ZMONTH   = 05
ZSTDYEAR = 2015
ZTIME    = 13:29
ZTIMEL   = 13:29:00.72
ZYEAR    = 15
ZUSER    = USER001
ZLOGON   = ESSUSER
ZSYSID   = TESTMVS
ZAPPLID  = ISR
ZDEL     = ;
ZENVIR   = ISPF 5.9MVS     TSO
ZLANG    = ENGLISH
ZPANELID = ISREDDE4
ZPREFIX  =
```

PREV
12 Tables – Create and edit

NEXT
14 Edit macros – Create and apply