Tech Series

MVS/QUICKREF®
The Key Solution to z/OS Problems

Displaying System and Session Values

Many times, individual TSO/ISPF users might have a need to know certain specific values related to either the system they are running on or their own TSO session. Additionally, internal ISPF settings, tables and defaults might need to be examined.

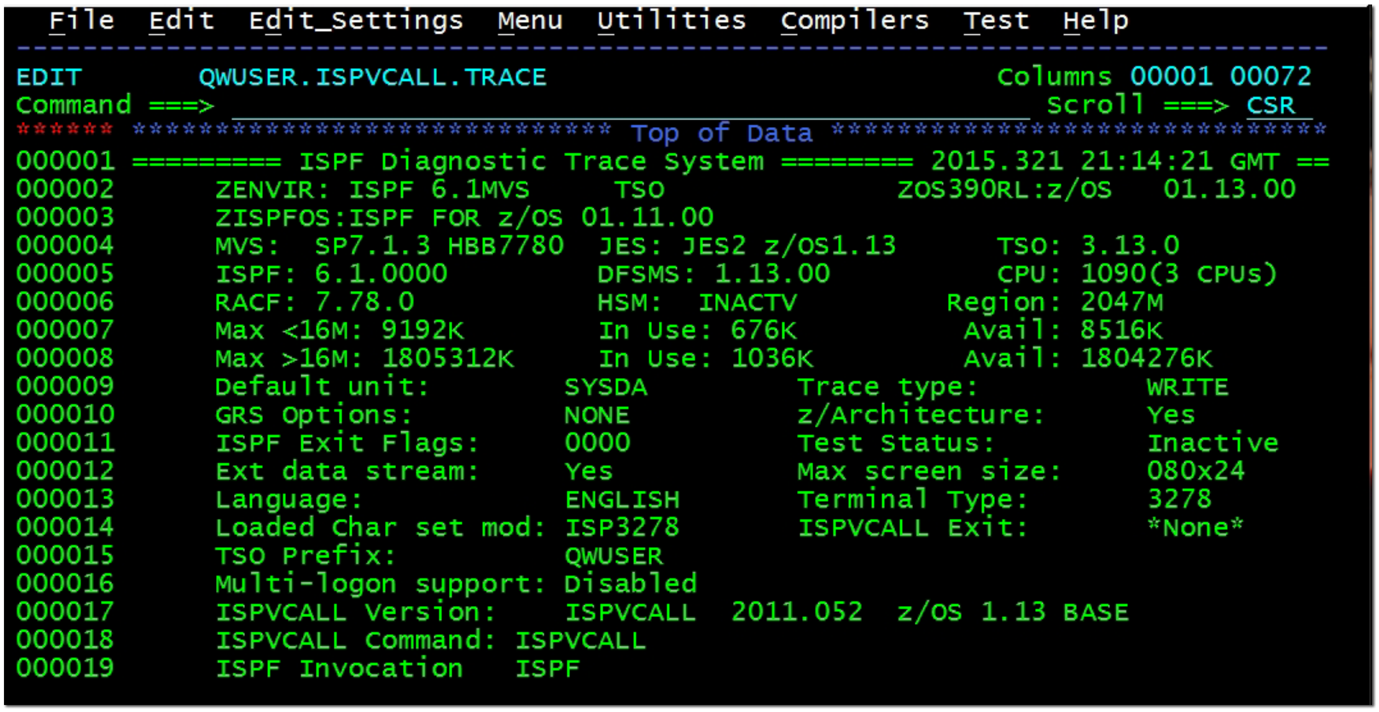
This article will discuss two methods of acquiring system/session level data. The first method is to use the handy ISPVCALL trace tool. The second method uses a simple, fall-through REXX EXEC that does nothing but use the MVSVAR and SYSVAR built-in functions to display values of global variables.

Both methods are valid, easy to use and will provide the needed views into low-level values and settings.

ISPVCALL

ISPVCALL provides many interesting details about a TSO/ISPF session. It may be invoked from ISPF Option 6. Executing it is as simple as entering: ISPVCALL STATUS from the Command line at Option 6.

Here's what the initial display looks like:


File Edit Edit_Settings Menu Utilities Compilers Test Help

```
EDIT       QWUSER.ISPVCALL.TRACE                          Columns 00001 00072
Command ===>                                                 Scroll ===> CSR
****** ***************************** Top of Data ******************************
000001 ========= ISPF Diagnostic Trace System ======== 2015.321 21:14:21 GMT ==
000002      ZENVIR: ISPF 6.1MVS     TSO             ZOS390RL:z/OS    01.13.00
000003      ZISPFOS:ISPF FOR z/OS 01.11.00
000004      MVS:  SP7.1.3 HBB7780  JES: JES2 z/OS1.13      TSO: 3.13.0
000005      ISPF: 6.1.0000         DFSMS: 1.13.00          CPU: 1090(3 CPUs)
000006      RACF: 7.78.0           HSM:  INACTV         Region: 2047M
000007      Max <16M: 9192K        In Use: 676K          Avail: 8516K
000008      Max >16M: 1805312K     In Use: 1036K         Avail: 1804276K
000009      Default unit:        SYSDA         Trace type:          WRITE
000010      GRS Options:         NONE          z/Architecture:      Yes
000011      ISPF Exit Flags:     0000          Test Status:         Inactive
000012      Ext data stream:     Yes           Max screen size:     080x24
000013      Language:            ENGLISH       Terminal Type:       3278
000014      Loaded Char set mod: ISP3278       ISPVCALL Exit:       *None*
000015      TSO Prefix:          QWUSER
000016      Multi-logon support: Disabled
000017      ISPVCALL Version:    ISPVCALL  2011.052  z/OS 1.13 BASE
000018      ISPVCALL Command: ISPVCALL
000019      ISPF Invocation    ISPF
```

continued on next page

Solutions... Not Just Software®

As you can see, there are many detailed fields displayed. On this initial screen, the displayed values are mostly system software release levels and other single-valued data items. This screen might be all you ever need to view if all you are interested in are release levels and other global z/OS settings and values.

However, by scrolling forward in the CALL.TRACE file, you will be able to see a stunning array of values, settings and other parameters that apply to your active TSO/ISPF session.

Even if you don't fully grasp the meanings of all of the information being displayed, it is nice to know that you DO have the means of answering some of your own questions about the internals of your environment.

SAYSYS REXX

For a number of years now, REXX programmers have been making use of the MVSVAR and SYSVAR built-in functions (BIFs). These particular BIFs are about as easy to use and understand as you'll ever see.

Their syntax is:

```
MVSVAR("variable")
```

or

```
SYSAR("variable")
```

continued next page

The only thing that is even remotely tricky about using them is knowing which named variables to use with MVSVAR and which to use with SYSVAR.

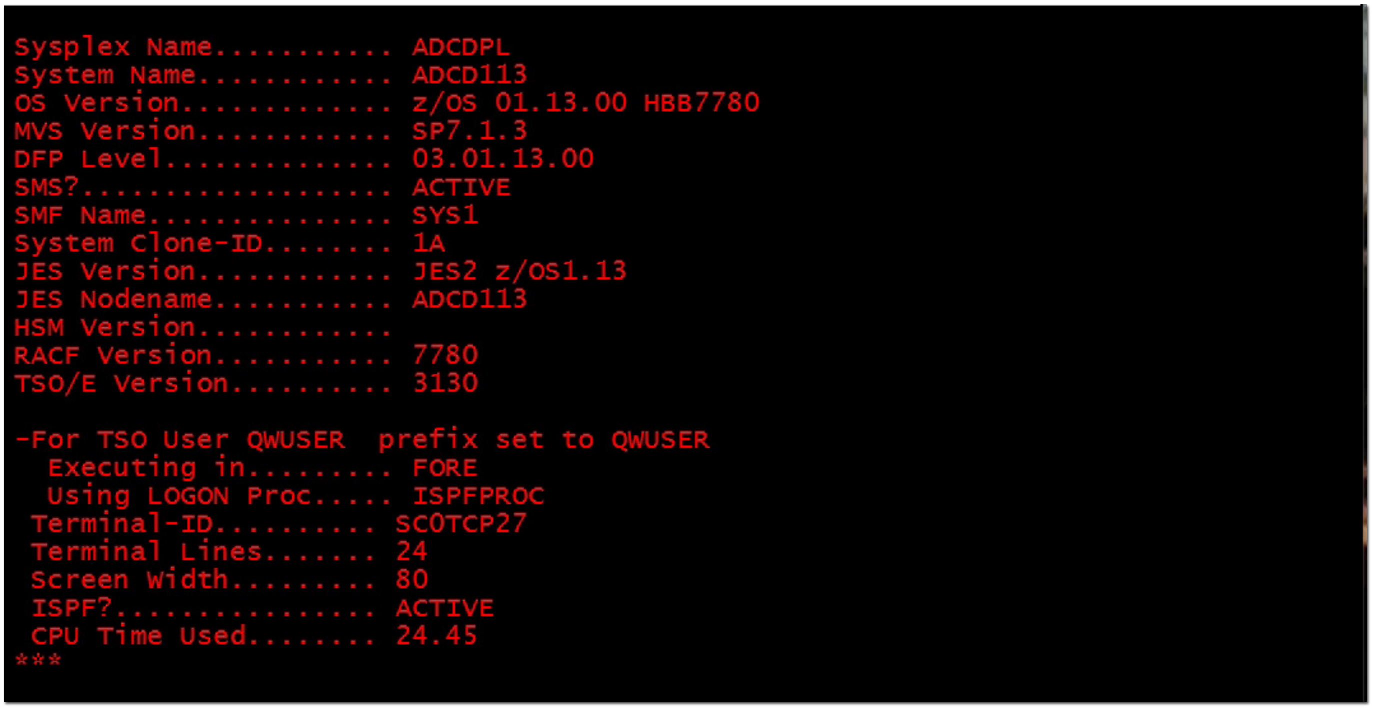
```
/* REXX */

Say ‘Sysplex Name...........’ Mvsvar(“SYSPLEX”)
Say ‘System Name............’ Mvsvar(“SYSNAME”)
Say ‘OS Version.............’ Mvsvar(“SYSOPSYS”)
Say ‘MVS Version............’ Mvsvar(“SYSMVS”)
Say ‘DFP Level..............’ Mvsvar(“SYSDFP”)
Say ‘SMS?...................’ Mvsvar(“SYSSMS”)
Say ‘SMF Name...............’ Mvsvar(“SYSSMFID”)
Say ‘System Clone-ID........’ Mvsvar(“SYSCLONE”)
Say ‘JES Version............’ Sysvar(“SYSJES”)
Say ‘JES Nodename...........’ Sysvar(“SYSNODE”)
Say ‘HSM Version............’ Sysvar(“SYSHSM”)
Say ‘RACF Version...........’ Sysvar(“SYSLRACF”)
Say ‘TSO/E Version..........’ Sysvar(“SYSTSOE”)
Say ‘ ‘
Say ‘-For TSO User’
Sysvar(“SYSUID”) ‘ prefix set to’ Sysvar(“SYSPREF”)
Say ‘ Executing in.........’ Sysvar(“SYSENV”)
Say ‘ Using LOGON Proc.....’ Sysvar(“SYSPROC”)
Say ‘ Terminal-ID..........’ Strip(Sysvar(“SYSTERMID”),’L’)
Say ‘ Terminal Lines.......’ Strip(Sysvar(“SYSLTERM”),’L’)
Say ‘ Screen Width.........’ Strip(Sysvar(“SYSWTERM”),’L’)
Say ‘ ISPF?................’ Sysvar(“SYSISPF”)
Say ‘ CPU Time Used........’ Strip(Sysvar(“SYSCPU”),’L’)
Say ‘ SRM Time Used........’ Strip(Sysvar(“SYSSRV”),’L’)
Exit
```

To use SAYSYS simply add it to a PDS in either the SYSPROC or SYSEXEC concatenation at your installation. Once in place, invoke it as:

```
Command ===> TSO SAYSYS
```

Here's what the output of the SAYSYS EXEC looks like:



```
Sysplex Name........... ADCDPL
System Name............ ADCD113
OS Version............. z/OS 01.13.00 HBB7780
MVS Version............ SP7.1.3
DFP Level.............. 03.01.13.00
SMS?................... ACTIVE
SMF Name............... SYS1
System Clone-ID........ 1A
JES Version............ JES2 z/OS1.13
JES Nodename........... ADCD113
HSM Version............
RACF Version........... 7780
TSO/E Version.......... 3130

-For TSO User QWUSER  prefix set to QWUSER
  Executing in......... FORE
  Using LOGON Proc..... ISPFPROC
 Terminal-ID.......... SCOTCP27
 Terminal Lines....... 24
 Screen Width......... 80
 ISPF?................ ACTIVE
 CPU Time Used........ 24.45
***
```

One of the niftier things that a REXX programmer can do with MVSVAR and SYSVAR values is to query them in order to take an appropriate action. For example, if a REXX EXEC is being written to run in both line-mode TSO AND full screen ISPF, the programmer could code something like this to determine how their REXX was running:

```
If Sysvar("SYSISPF") = "ACTIVE" Then
Say "Yup, ISPF baby!"
```

Conclusion

So there you have it. Two methods for peeking into the guts of your environment. One, ISPVCALL, is all pre-written and ready to use. Two, SAYSYS, also pre-written but open to your own experimentation and programming.

A final thought: Use MVS/Quick-Ref® to search for MVSVAR and SYSVAR to see the large number of values that are available to you as "variables" in the MVSVAR / SYSVAR BIFs.