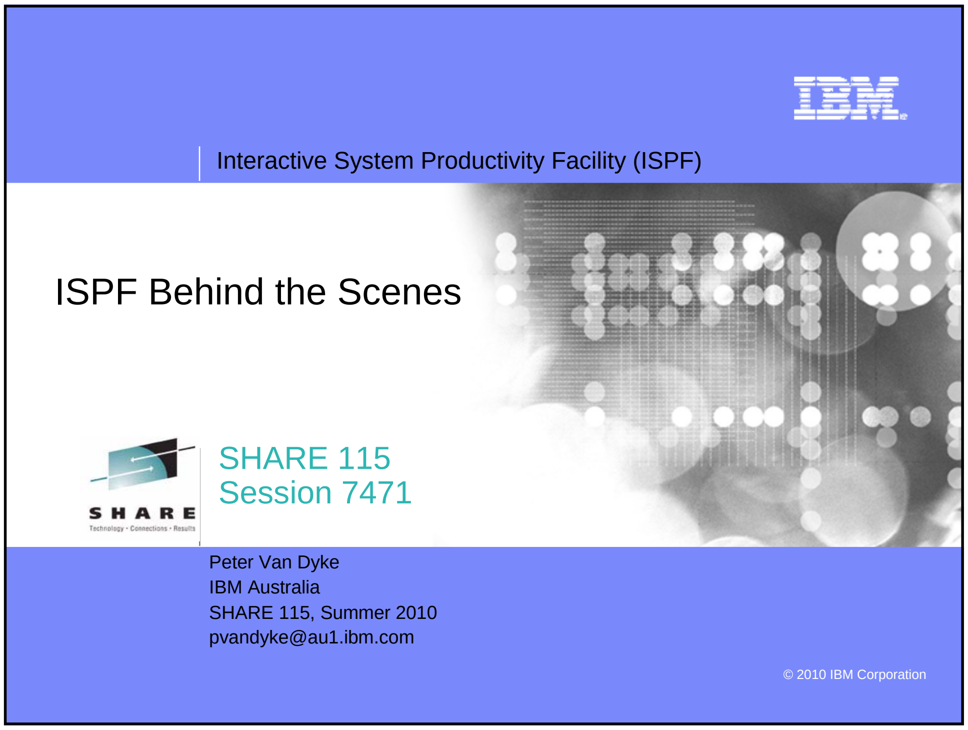
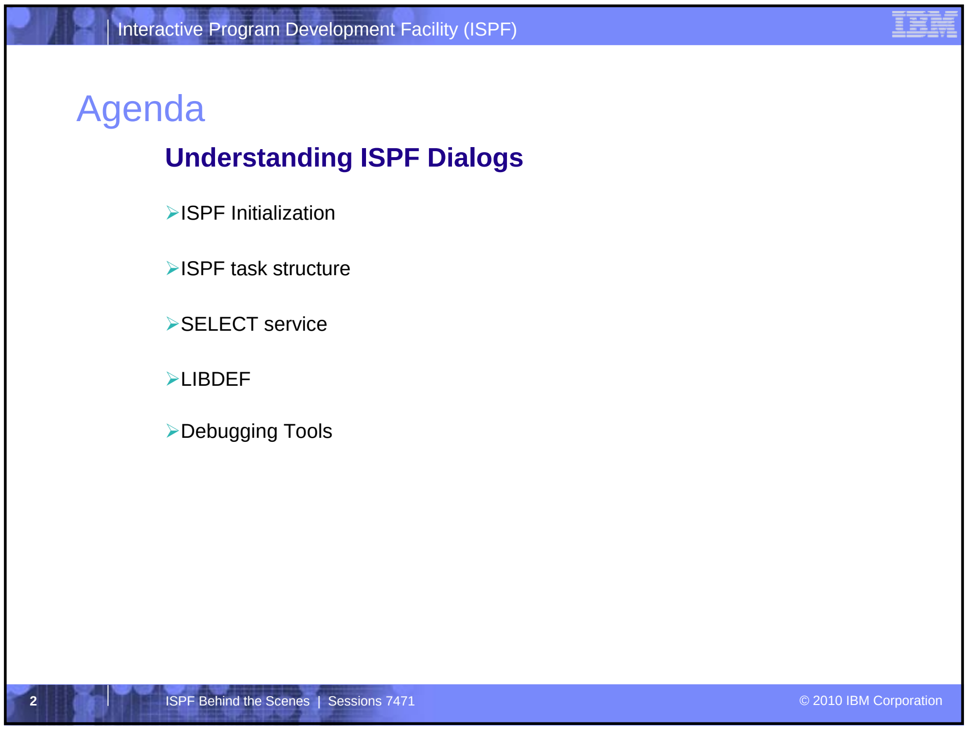


Interactive System Productivity Facility (ISPF)

ISPF Behind the Scenes

SHARE 115
Session 7471

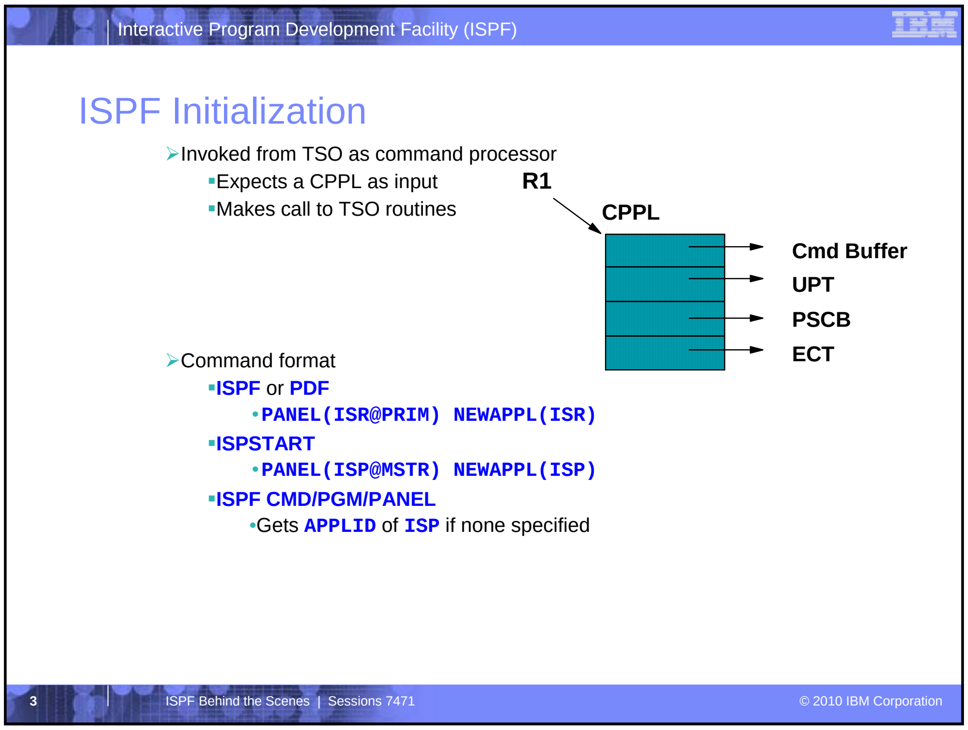
Peter Van Dyke
IBM Australia
SHARE 115, Summer 2010
pvandyke@au1.ibm.com



Agenda

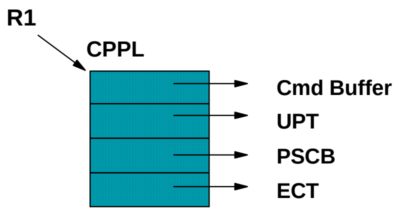
Understanding ISPF Dialogs

- ISPF Initialization
- ISPF task structure
- SELECT service
- LIBDEF
- Debugging Tools

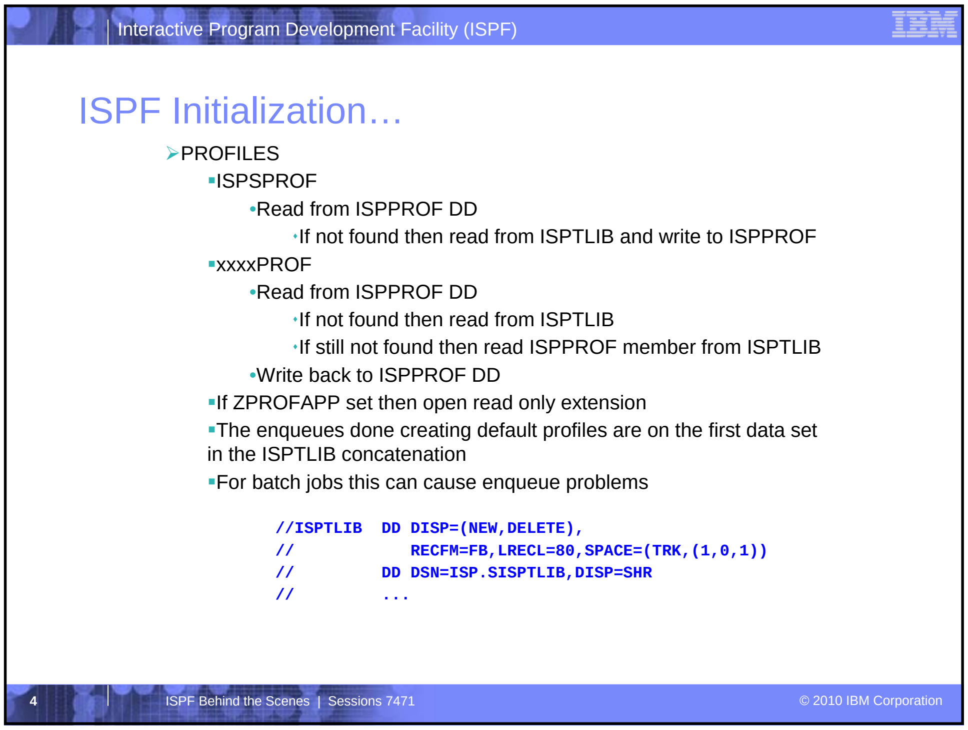


ISPF Initialization

- ➢Invoked from TSO as command processor
  - ▪Expects a CPPL as input
  - ▪Makes call to TSO routines



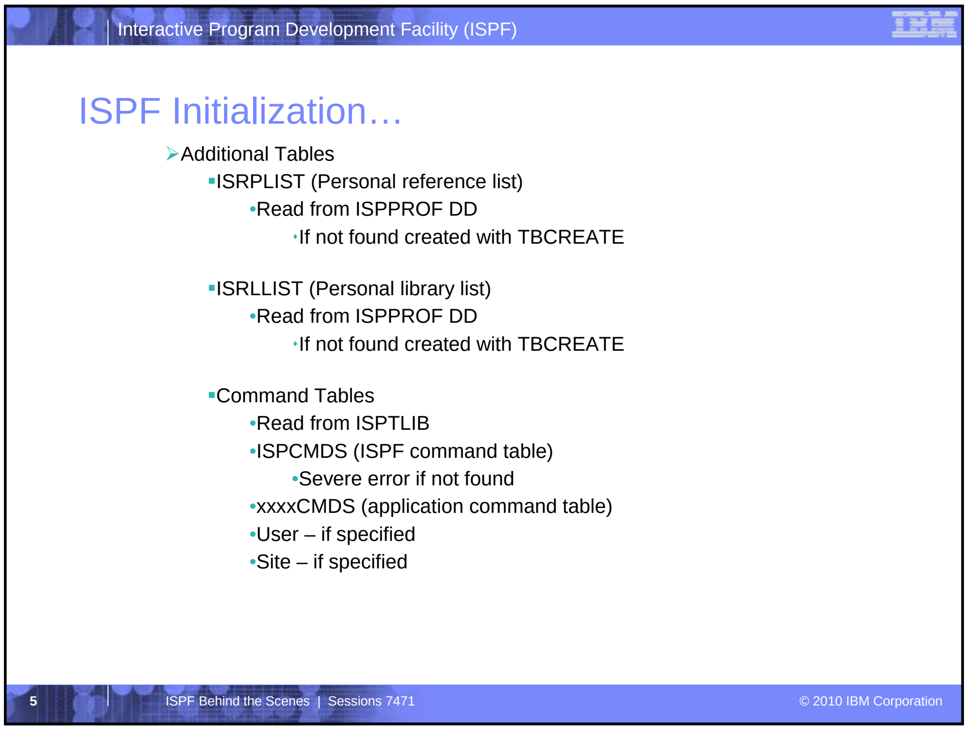
- ➢Command format
  - ▪**ISPF** or **PDF**
    - •**`PANEL(ISR@PRIM) NEWAPPL(ISR)`**
  - ▪**ISPSTART**
    - •**`PANEL(ISP@MSTR) NEWAPPL(ISP)`**
  - ▪**ISPF CMD/PGM/PANEL**
    - •Gets **`APPLID`** of **`ISP`** if none specified



ISPF Initialization…

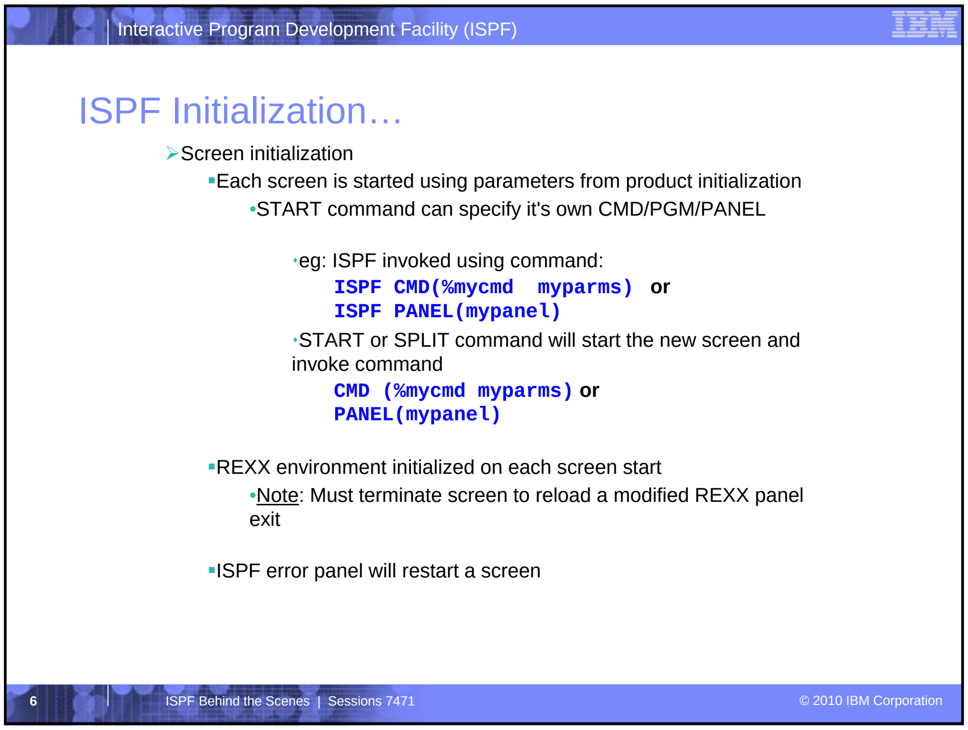
- ➢PROFILES
  - ▪ISPSPROF
    - •Read from ISPPROF DD
      - •If not found then read from ISPTLIB and write to ISPPROF
  - ▪xxxxPROF
    - •Read from ISPPROF DD
      - •If not found then read from ISPTLIB
      - •If still not found then read ISPPROF member from ISPTLIB
    - •Write back to ISPPROF DD
  - ▪If ZPROFAPP set then open read only extension
  - ▪The enqueues done creating default profiles are on the first data set in the ISPTLIB concatenation
  - ▪For batch jobs this can cause enqueue problems

```
//ISPTLIB  DD DISP=(NEW,DELETE),
//            RECFM=FB,LRECL=80,SPACE=(TRK,(1,0,1))
//         DD DSN=ISP.SISPTLIB,DISP=SHR
//         ...
```



ISPF Initialization…

- Additional Tables
  - ISRPLIST (Personal reference list)
    - Read from ISPPROF DD
      - If not found created with TBCREATE
  - ISRLLIST (Personal library list)
    - Read from ISPPROF DD
      - If not found created with TBCREATE
  - Command Tables
    - Read from ISPTLIB
    - ISPCMDS (ISPF command table)
      - Severe error if not found
    - xxxxCMDS (application command table)
    - User – if specified
    - Site – if specified



ISPF Initialization…

- Screen initialization
  - Each screen is started using parameters from product initialization
    - START command can specify it's own CMD/PGM/PANEL
      - eg: ISPF invoked using command:

        `ISPF CMD(%mycmd  myparms)` **or**
        `ISPF PANEL(mypanel)`
      - START or SPLIT command will start the new screen and invoke command

        `CMD (%mycmd myparms)` **or**
        `PANEL(mypanel)`
  - REXX environment initialized on each screen start
    - <u>Note</u>: Must terminate screen to reload a modified REXX panel exit
  - ISPF error panel will restart a screen

KEYLISTS

- PF Key definitions associated with panels

- Coded with)PANEL statement
  - **)PANEL [KEYLIST(keylist name,[applid,[SHARED]])]**
    - ISPKYLST is used if keylist not specified

- Normally in xxxxKEYS table in ISPTLIB
  - Use DTL to create

```
<!DOCTYPE DM SYSTEM>
<KEYL NAME=MYKEYLST APPLID=ABC>
  <KEYI KEY=F1  CMD=HELP   FKA=YES>Help
  <KEYI KEY=F2  CMD=SPLIT  FKA=LONG>Split
  <KEYI KEY=F3  CMD=EXIT   FKA=YES>Exit
  ...
  <KEYI KEY=F24 CMD=CANCEL FKA=YES>Cancel
</KEYL>
```

- Private copies
  - Created by Keylist Utility
  - Stored in xxxxPROF
  - Stored in ISPSPROF for applid ISP

KEYLISTS…

- KEYLIST Search

  - Keylists ignored if profile variable **ZKLUSE=N**
    (unless SHARED specified)

  - SHARED
    1. xxxxKEYS
    2. ISPKEYS

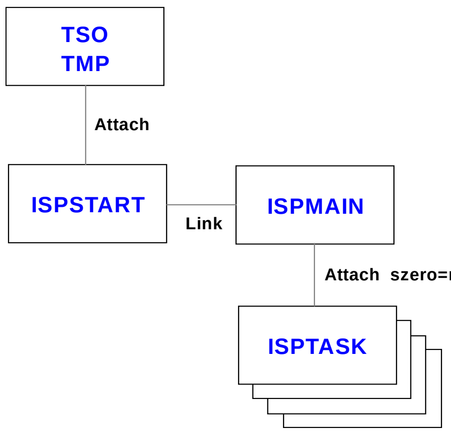
  - SHARED not specified
    1. xxxxPROF
    2. xxxxKEYS
    3. ISPSPROF
    4. ISPKEYS

  - ISPF uses the current APPLID if no applid is specified

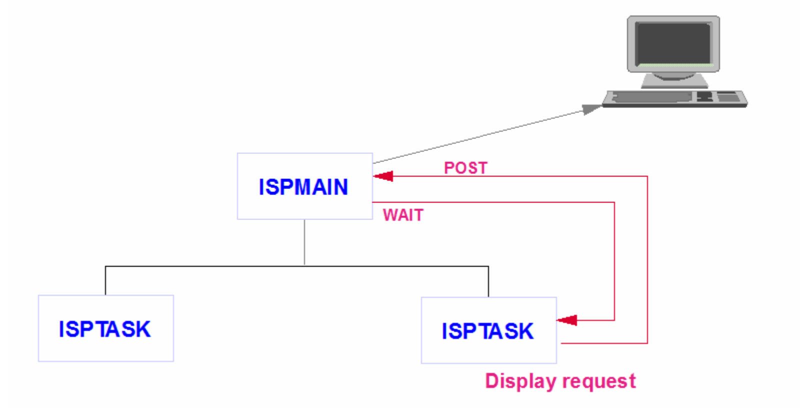
ISPF Services

- All ISPF services are run from ISPTASK TCB
- Interface:
  - ISPLINK
    `eg: CALL ISPLINK('SETMSG  ','ISPZ001 ',' ','ZCMD')`
  - ISPEXEC
    `eg: ISPEXEC SETMSG MSG(ISPZ001) MSGLOC(ZCMD)`
- ISPLINK parameters
  - Positional
  - To omit a parameter and use default - code a blank
    - Note: An address starting with x'40' will be treated as an omitted parameter.
  - Last address in parameter list must have high order bit on
    - Use the VL keyword in Assembler call statements
  - Standard linkage conventions are observed
  - Keywords and names should be padded to the max length of 8
  - Numeric values are full word binary
    - Don't rely on coding constant. Compiler may not generate a full word value.

ISPF Task Structure



ISPF Task Structure…

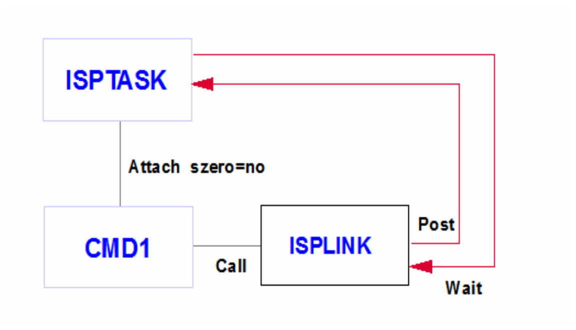


ISPF Task Structure…

- DISPLAY of panel
  - Actual display is done by ISPMAIN task with SVC 93 (TPUT/TGET)
  - ISPMAIN will normally be in wait state waiting for user to press enter key
  - ISPTASK is in wait

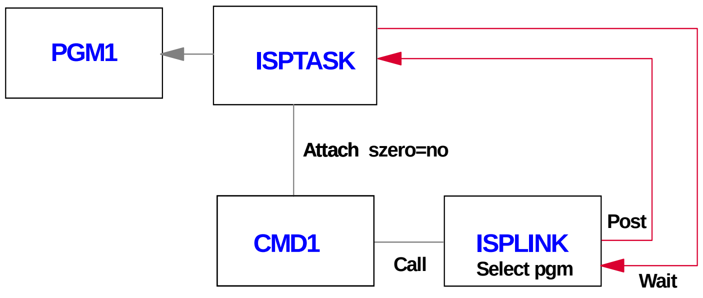
ISPF Task Structure…

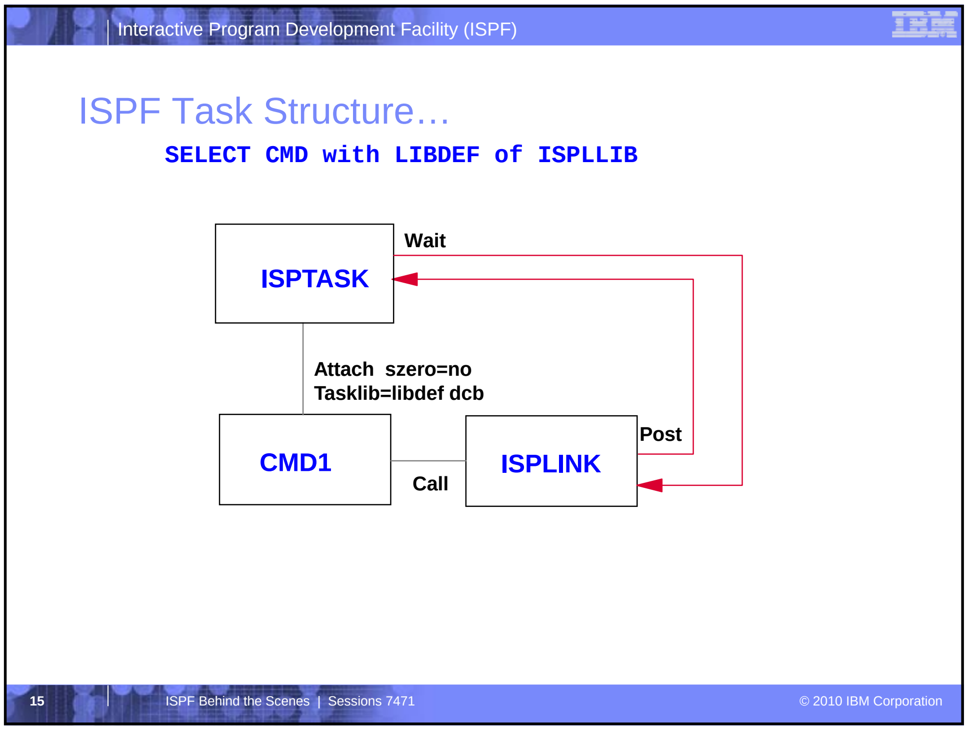
SELECT CMD



ISPF Task Structure…

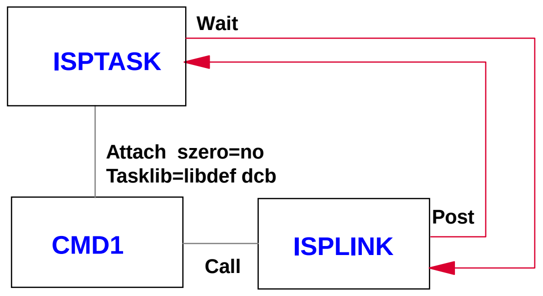
SELECT CMD invoking SELECT PGM

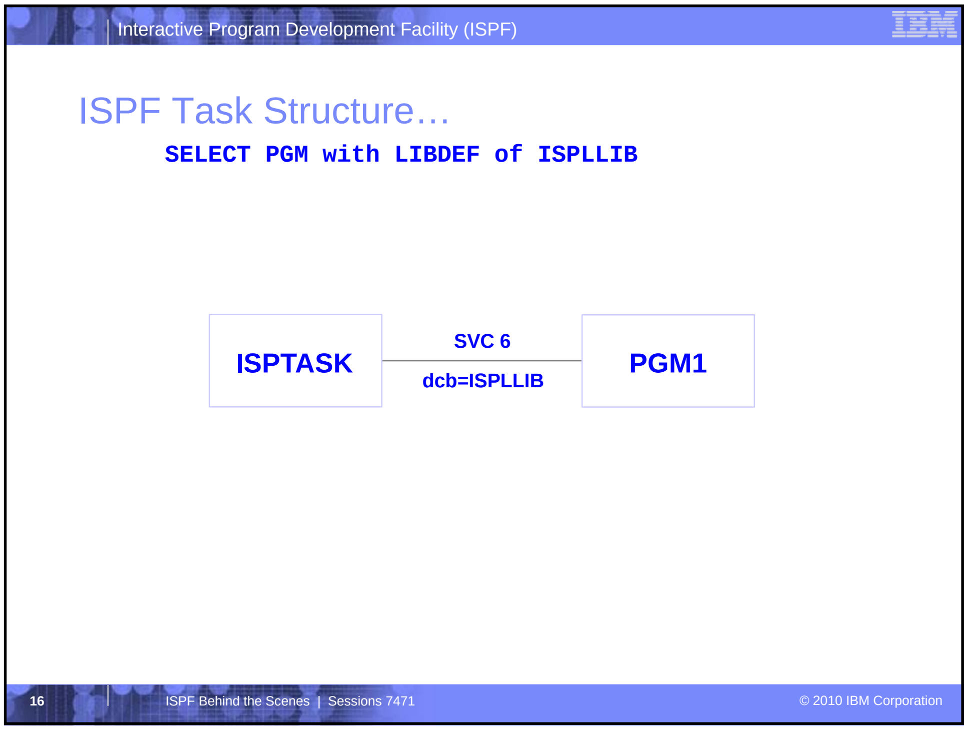




ISPF Task Structure…

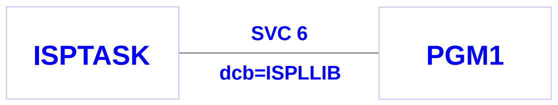
SELECT CMD with LIBDEF of ISPLLIB

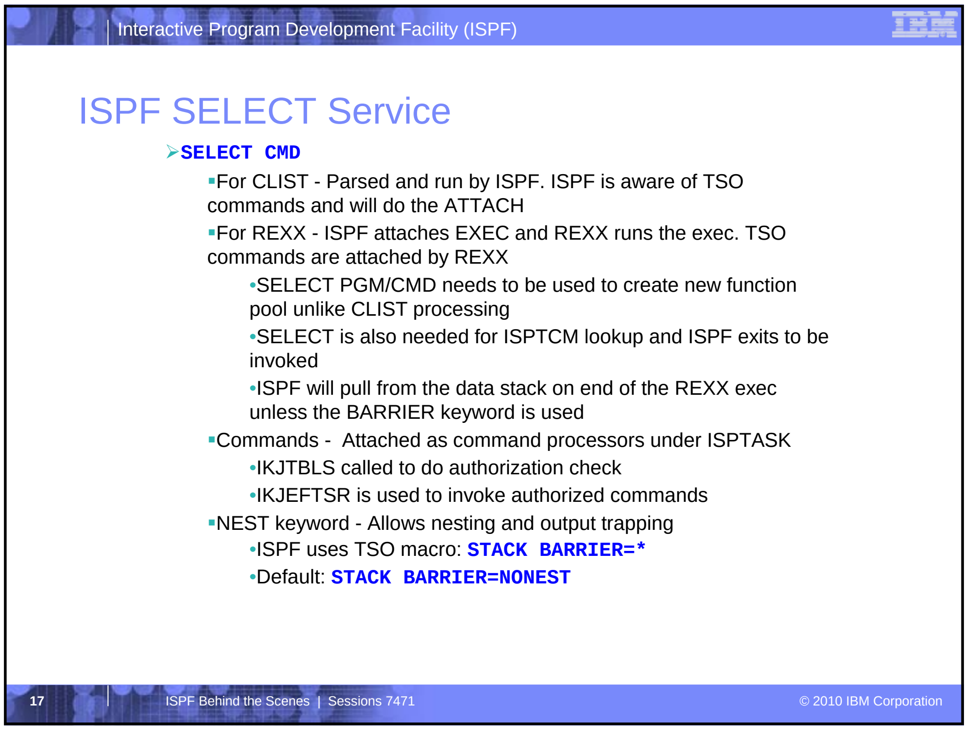




ISPF Task Structure…

SELECT PGM with LIBDEF of ISPLLIB

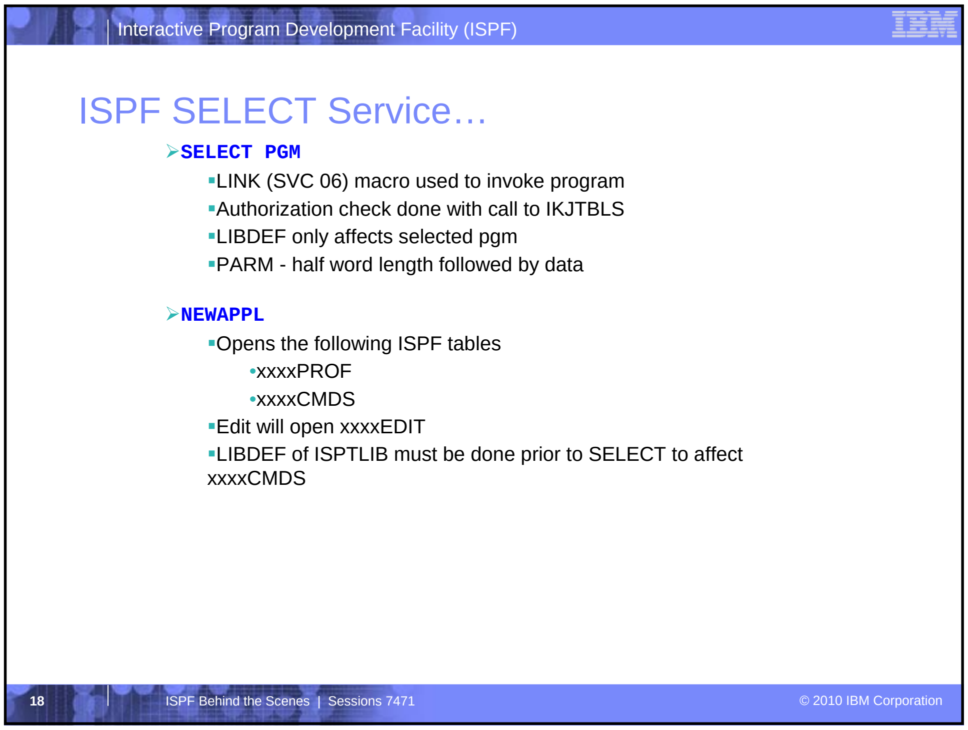




ISPF SELECT Service

➢**SELECT CMD**

- For CLIST - Parsed and run by ISPF. ISPF is aware of TSO commands and will do the ATTACH
- For REXX - ISPF attaches EXEC and REXX runs the exec. TSO commands are attached by REXX
  - SELECT PGM/CMD needs to be used to create new function pool unlike CLIST processing
  - SELECT is also needed for ISPTCM lookup and ISPF exits to be invoked
  - ISPF will pull from the data stack on end of the REXX exec unless the BARRIER keyword is used
- Commands - Attached as command processors under ISPTASK
  - IKJTBLS called to do authorization check
  - IKJEFTSR is used to invoke authorized commands
- NEST keyword - Allows nesting and output trapping
  - ISPF uses TSO macro: **`STACK BARRIER=*`**
  - Default: **`STACK BARRIER=NONEST`**



ISPF SELECT Service…

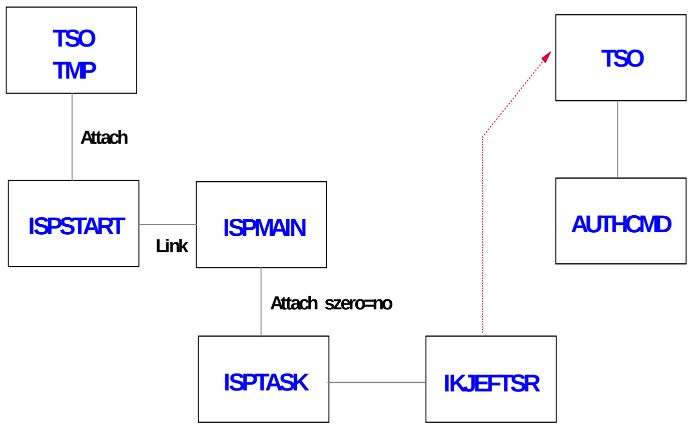
➢**SELECT PGM**

- LINK (SVC 06) macro used to invoke program
- Authorization check done with call to IKJTBLS
- LIBDEF only affects selected pgm
- PARM - half word length followed by data

➢**NEWAPPL**

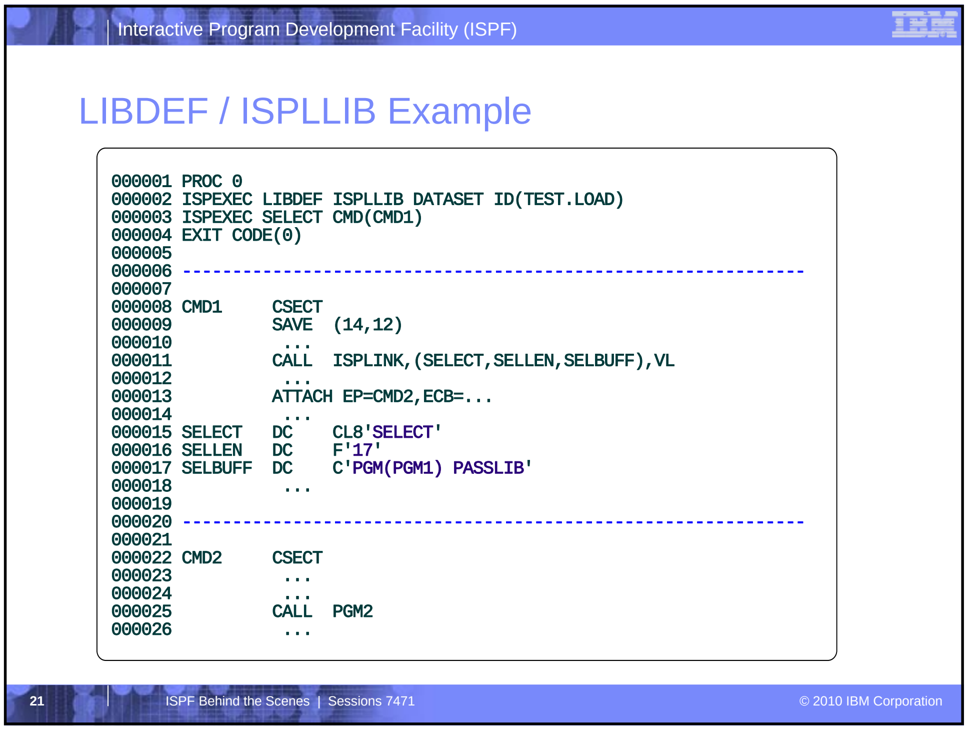
- Opens the following ISPF tables
  - xxxxPROF
  - xxxxCMDS
- Edit will open xxxxEDIT
- LIBDEF of ISPTLIB must be done prior to SELECT to affect xxxxCMDS

ISPF SELECT Authorized Commands



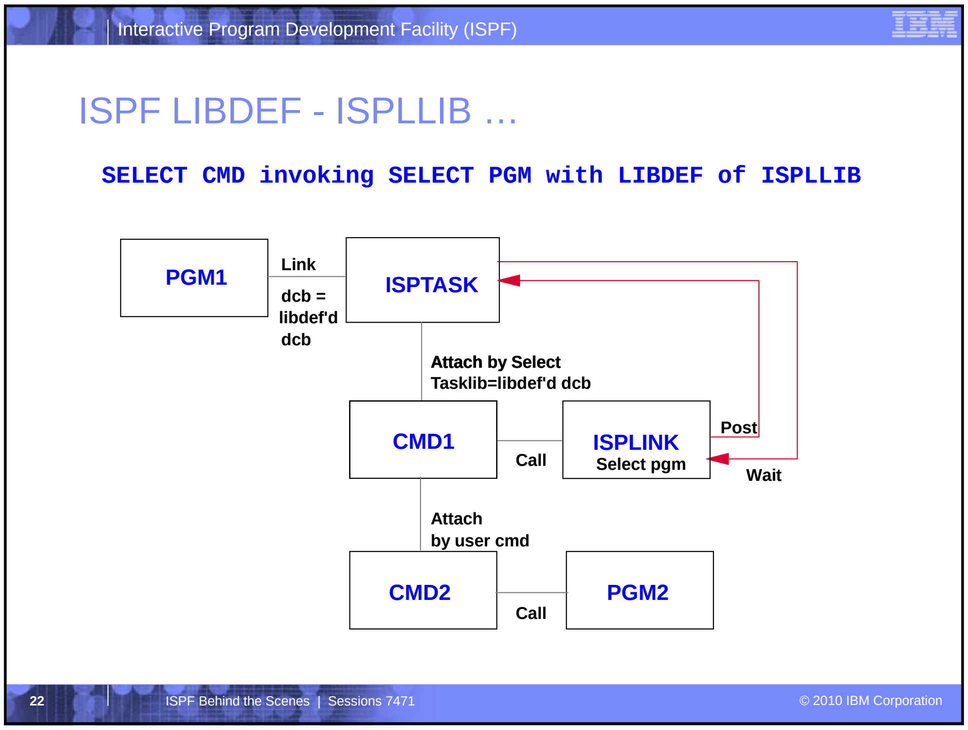
ISPF LIBDEF - ISPLLIB

- ISPLLIB
  - Used to pick up ISPF modules on product initialization
  - On invocation of ISPF it is used as TASKLIB to start an ISPF screen
- LIBDEF of ISPLLIB
  - **`SELECT PGM`**
    - DCB parm on LINK macro used to point to user load library
    - DCB parm only affect the module that LINK invokes
    - If EXCLDATA/EXCLLIBR used:
      LINK with DCB=*LIBDEFed dcb*
    - Otherwise BLDL is done on libdef'd DCB
      - BLDL finds module:
        LINK with DCB=*LIBDEFed dcb*
      - BLDL doesn't find module:
        - LINK with DCB=0
  - **`SELECT CMD`**
    - ATTACH of command is done with TASKLIB=*LIBDEFed dcb*



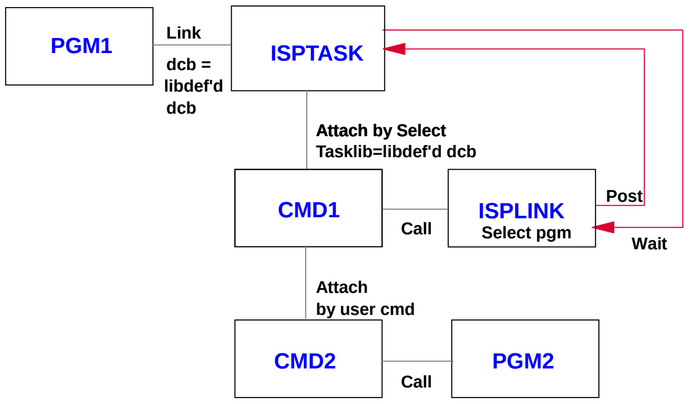
LIBDEF / ISPLLIB Example

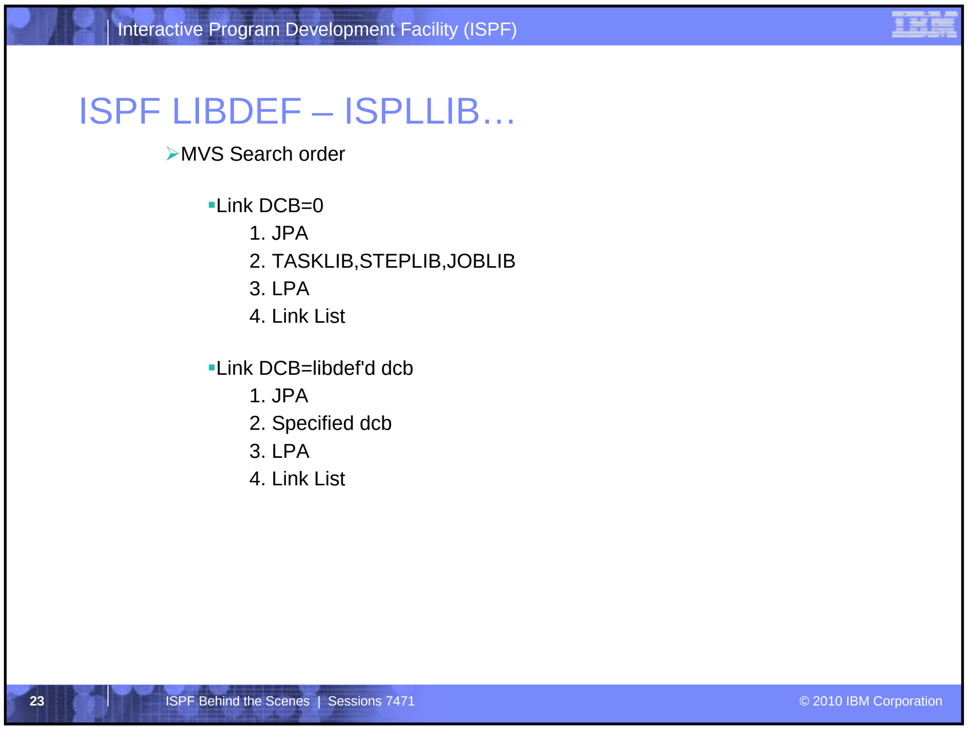
```
PROC 0
ISPEXEC LIBDEF ISPLLIB DATASET ID(TEST.LOAD)
ISPEXEC SELECT CMD(CMD1)
EXIT CODE(0)

---------------------------------------------------------------

CMD1     CSECT
         SAVE  (14,12)
          ...
         CALL  ISPLINK,(SELECT,SELLEN,SELBUFF),VL
          ...
         ATTACH EP=CMD2,ECB=...
          ...
SELECT   DC    CL8'SELECT'
SELLEN   DC    F'17'
SELBUFF  DC    C'PGM(PGM1) PASSLIB'
          ...

---------------------------------------------------------------

CMD2     CSECT
          ...
          ...
         CALL  PGM2
          ...
```



ISPF LIBDEF - ISPLLIB …

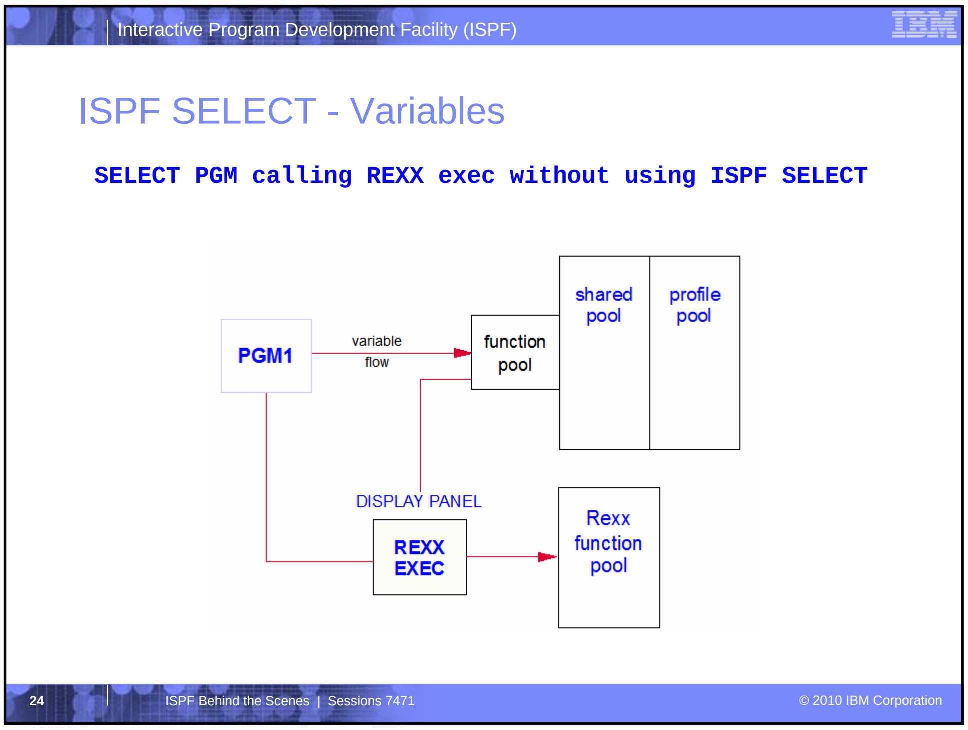
SELECT CMD invoking SELECT PGM with LIBDEF of ISPLLIB





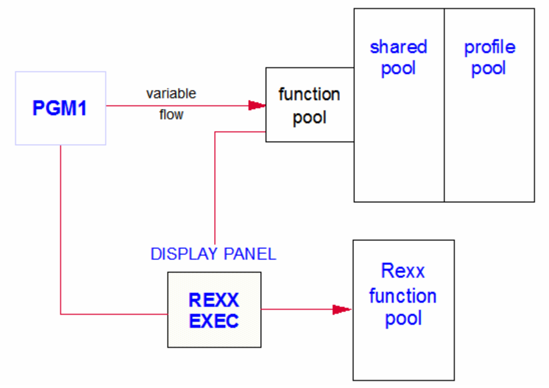
ISPF LIBDEF – ISPLLIB…

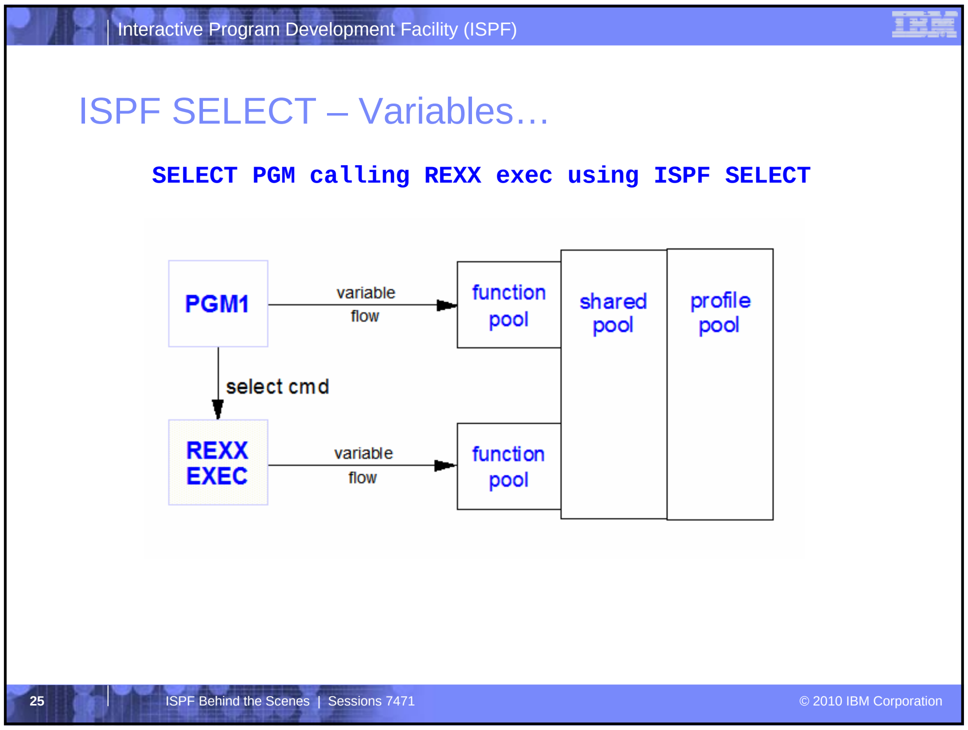
- MVS Search order
  - Link DCB=0
    1. JPA
    2. TASKLIB,STEPLIB,JOBLIB
    3. LPA
    4. Link List
  - Link DCB=libdef'd dcb
    1. JPA
    2. Specified dcb
    3. LPA
    4. Link List



ISPF SELECT - Variables

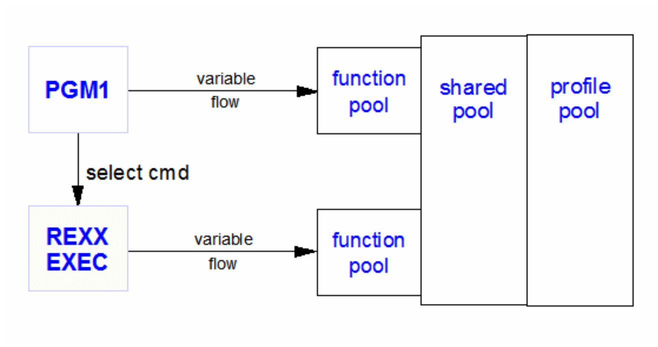
SELECT PGM calling REXX exec without using ISPF SELECT

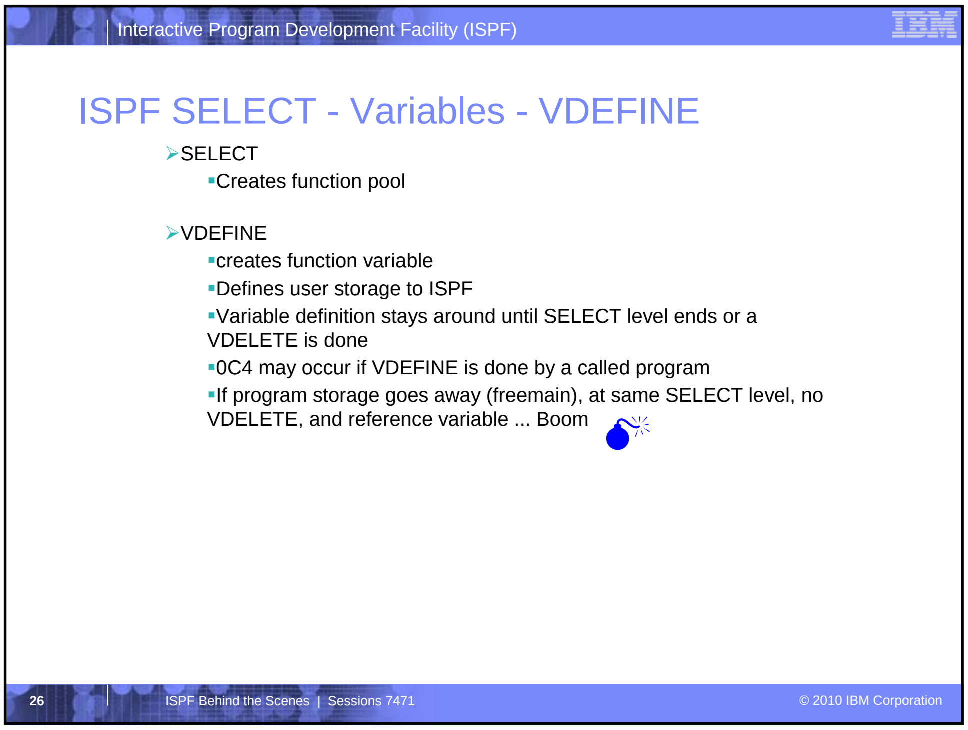




ISPF SELECT – Variables…

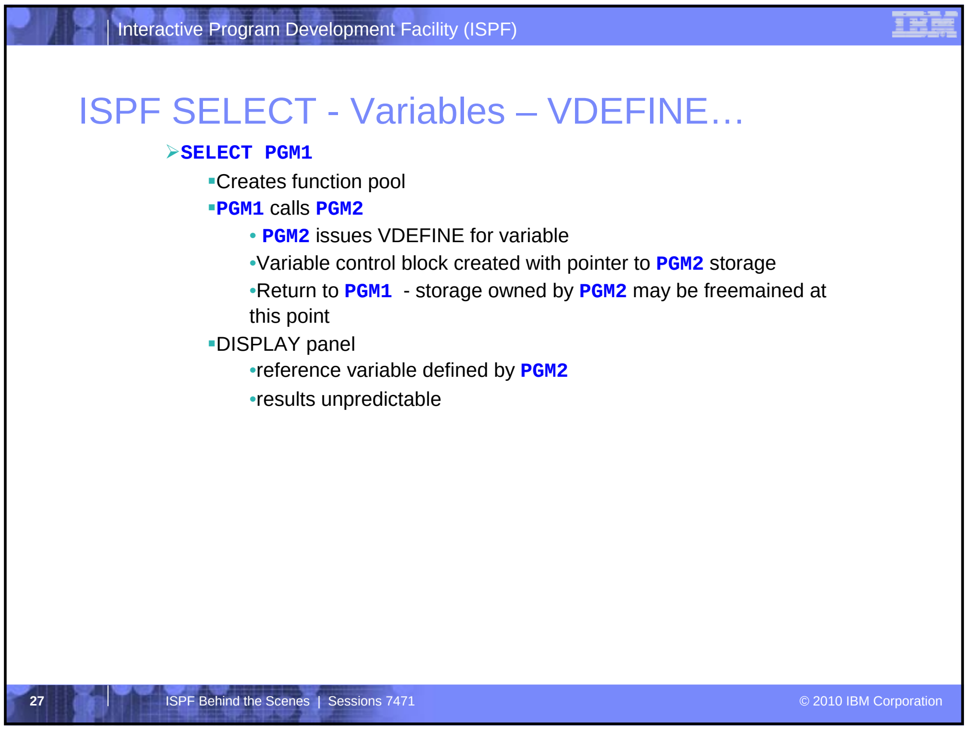
SELECT PGM calling REXX exec using ISPF SELECT





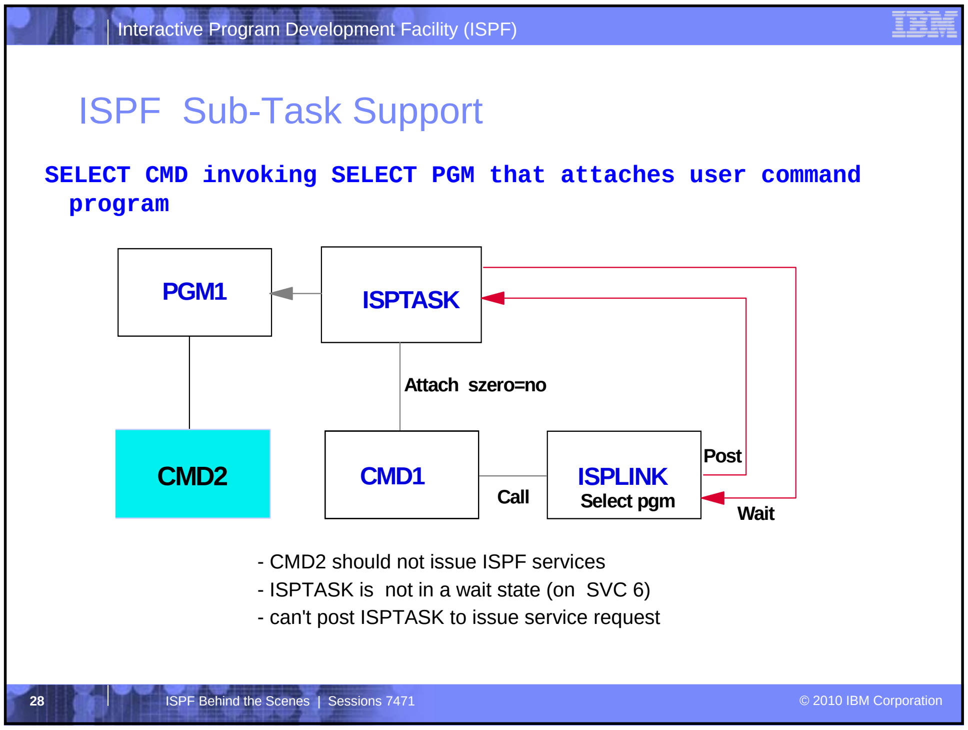
ISPF SELECT - Variables - VDEFINE

- SELECT
  - Creates function pool
- VDEFINE
  - creates function variable
  - Defines user storage to ISPF
  - Variable definition stays around until SELECT level ends or a VDELETE is done
  - 0C4 may occur if VDEFINE is done by a called program
  - If program storage goes away (freemain), at same SELECT level, no VDELETE, and reference variable ... Boom



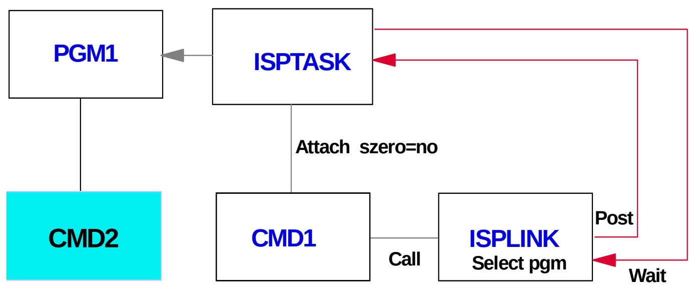
ISPF SELECT - Variables – VDEFINE…

- **SELECT PGM1**
  - Creates function pool
  - **PGM1** calls **PGM2**
    - **PGM2** issues VDEFINE for variable
    - Variable control block created with pointer to **PGM2** storage
    - Return to **PGM1** - storage owned by **PGM2** may be freemained at this point
  - DISPLAY panel
    - reference variable defined by **PGM2**
    - results unpredictable



ISPF Sub-Task Support

SELECT CMD invoking SELECT PGM that attaches user command program



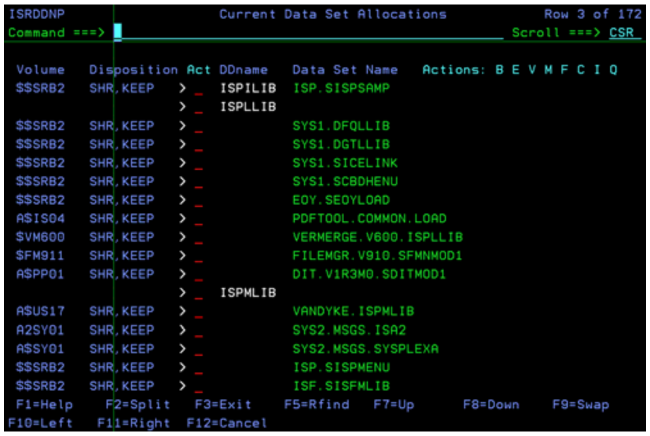
- CMD2 should not issue ISPF services
- ISPTASK is not in a wait state (on SVC 6)
- can't post ISPTASK to issue service request

ISPF Debugging Tools

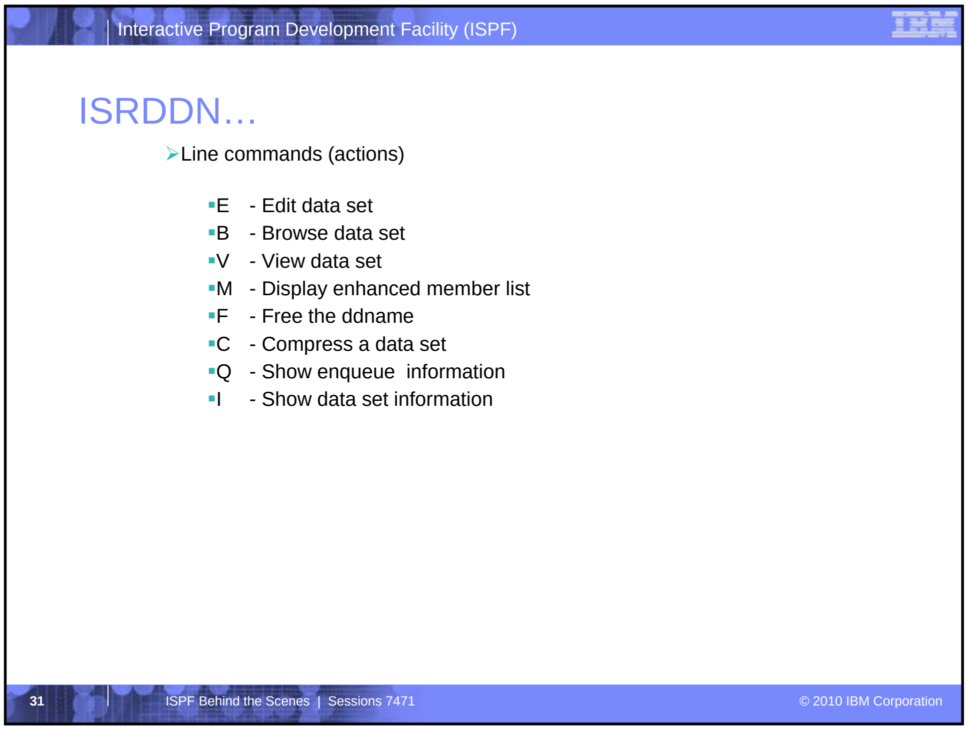
- ➢ISRDDN
- ➢ISPVCALL
- ➢Dialog Test
- ➢Other

ISRDDN

- ➢Scrollable list of allocated DD's and associated data set names
- ➢Invoked using TSO ISRDDN or DDLIST commands
- ➢Documented in the ISPF User’s Guide Volume I

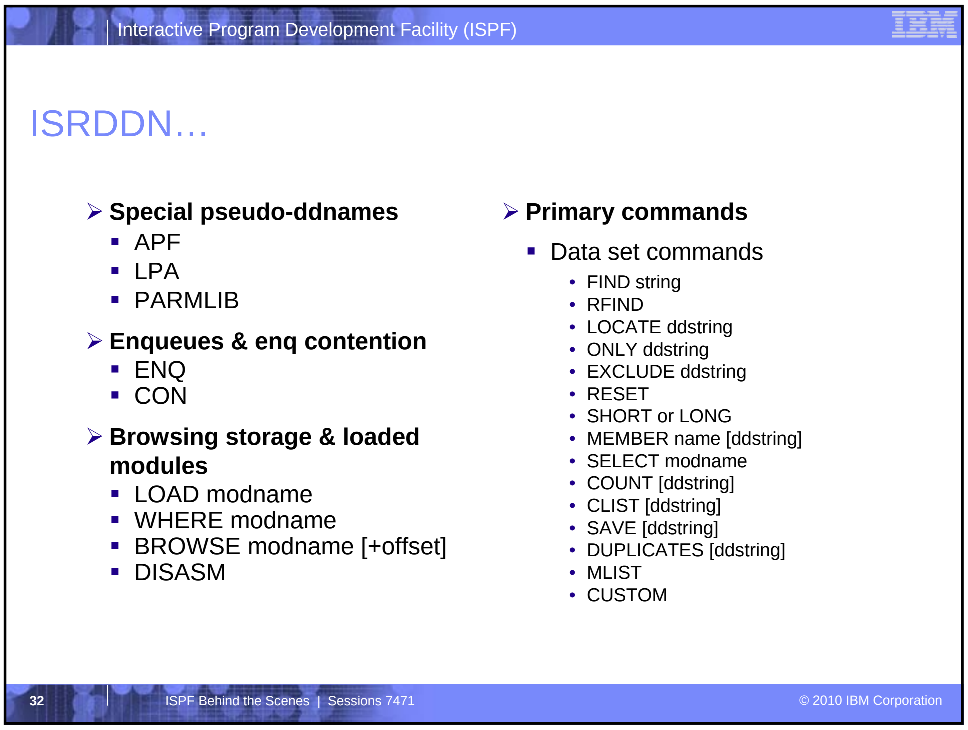


```
ISRDDNP                    Current Data Set Allocations          Row 3 of 172
Command ===>                                                    Scroll ===> CSR

 Volume   Disposition Act DDname   Data Set Name    Actions: B E V M F C I Q
 $$SRB2   SHR,KEEP    >  _ ISPILIB  ISP.SISPSAMP
                      >  _ ISPLLIB
 $$SRB2   SHR,KEEP    >  _          SYS1.DFQLLIB
 $$SRB2   SHR,KEEP    >  _          SYS1.DGTLLIB
 $$SRB2   SHR,KEEP    >  _          SYS1.SICELINK
 $$SRB2   SHR,KEEP    >  _          SYS1.SCBDHENU
 $$SRB2   SHR,KEEP    >  _          EOY.SEOYLOAD
 A$IS04   SHR,KEEP    >  _          PDFTOOL.COMMON.LOAD
 $VM600   SHR,KEEP    >  _          VERMERGE.V600.ISPLLIB
 $FM911   SHR,KEEP    >  _          FILEMGR.V910.SFMNMOD1
 A$PP01   SHR,KEEP    >  _          DIT.V1R3M0.SDITMOD1
                      >  _ ISPMLIB
 A$US17   SHR,KEEP    >  _          VANDYKE.ISPMLIB
 A2SY01   SHR,KEEP    >  _          SYS2.MSGS.ISA2
 A$SY01   SHR,KEEP    >  _          SYS2.MSGS.SYSPLEXA
 $$SRB2   SHR,KEEP    >  _          ISP.SISPMENU
 $$SRB2   SHR,KEEP    >  _          ISF.SISFMLIB
 F1=Help     F2=Split    F3=Exit     F5=Rfind    F7=Up       F8=Down     F9=Swap
F10=Left    F11=Right   F12=Cancel
```



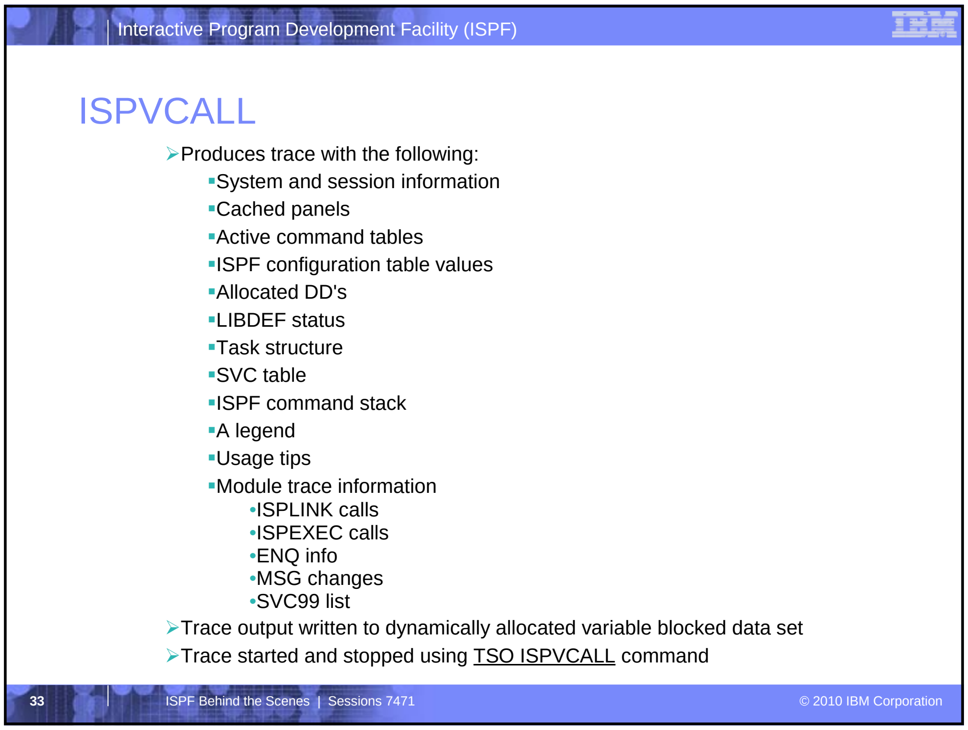
ISRDDN…

- Line commands (actions)
  - E - Edit data set
  - B - Browse data set
  - V - View data set
  - M - Display enhanced member list
  - F - Free the ddname
  - C - Compress a data set
  - Q - Show enqueue information
  - I - Show data set information



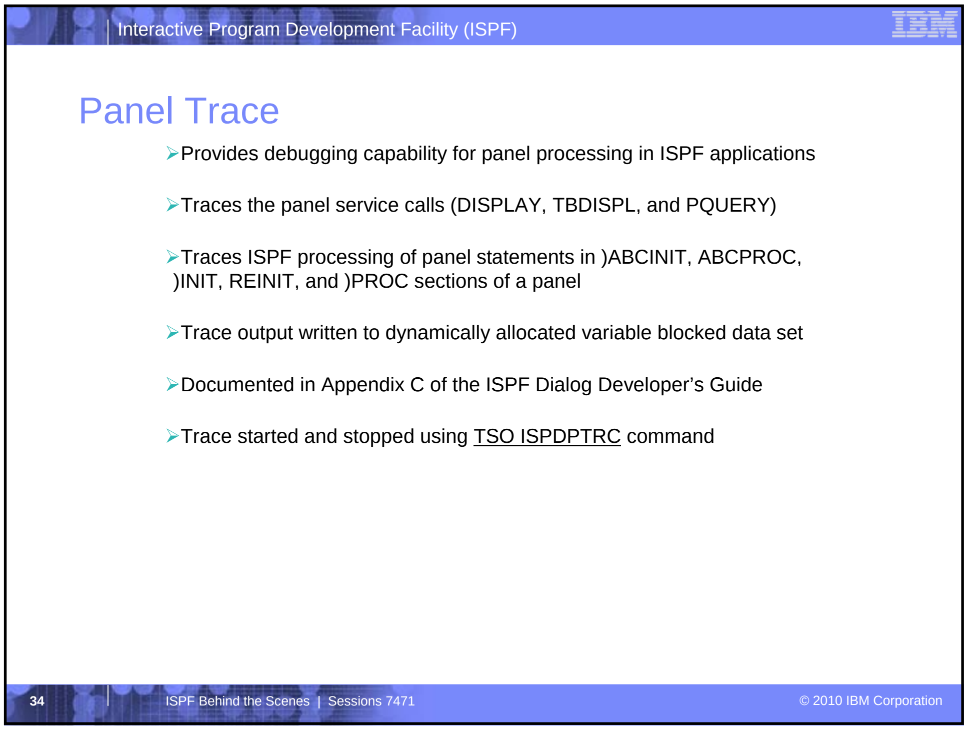
ISRDDN…

- **Special pseudo-ddnames**
  - APF
  - LPA
  - PARMLIB
- **Enqueues & enq contention**
  - ENQ
  - CON
- **Browsing storage & loaded modules**
  - LOAD modname
  - WHERE modname
  - BROWSE modname [+offset]
  - DISASM
- **Primary commands**
  - Data set commands
    - FIND string
    - RFIND
    - LOCATE ddstring
    - ONLY ddstring
    - EXCLUDE ddstring
    - RESET
    - SHORT or LONG
    - MEMBER name [ddstring]
    - SELECT modname
    - COUNT [ddstring]
    - CLIST [ddstring]
    - SAVE [ddstring]
    - DUPLICATES [ddstring]
    - MLIST
    - CUSTOM



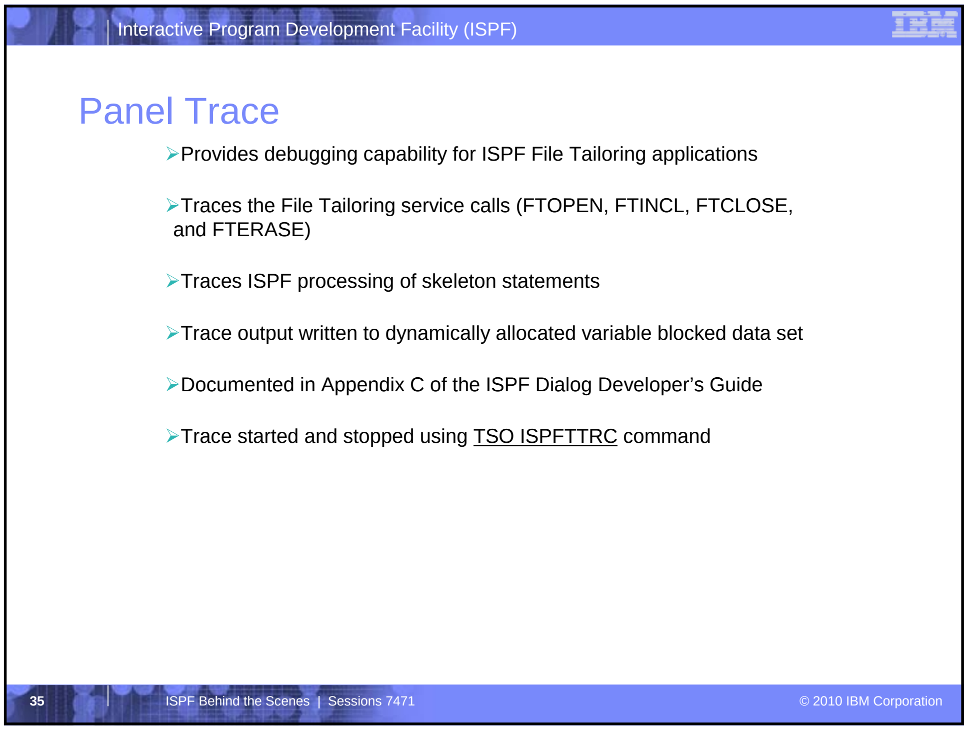
ISPVCALL

- Produces trace with the following:
  - System and session information
  - Cached panels
  - Active command tables
  - ISPF configuration table values
  - Allocated DD's
  - LIBDEF status
  - Task structure
  - SVC table
  - ISPF command stack
  - A legend
  - Usage tips
  - Module trace information
    - ISPLINK calls
    - ISPEXEC calls
    - ENQ info
    - MSG changes
    - SVC99 list
- Trace output written to dynamically allocated variable blocked data set
- Trace started and stopped using TSO ISPVCALL command



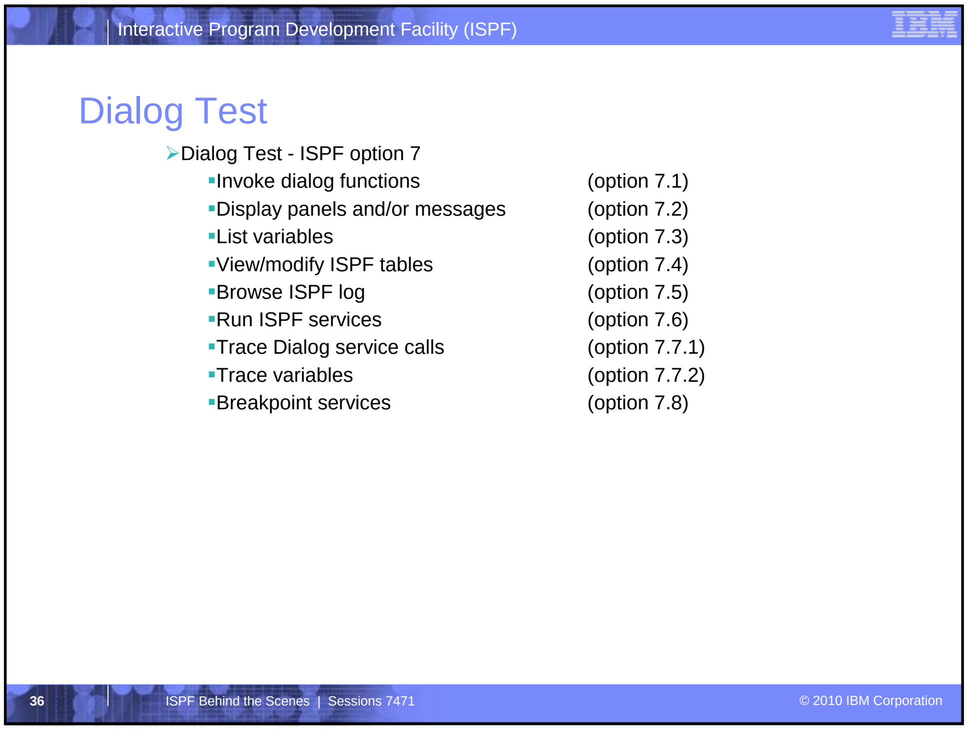
Panel Trace

- Provides debugging capability for panel processing in ISPF applications
- Traces the panel service calls (DISPLAY, TBDISPL, and PQUERY)
- Traces ISPF processing of panel statements in)ABCINIT, ABCPROC,)INIT, REINIT, and)PROC sections of a panel
- Trace output written to dynamically allocated variable blocked data set
- Documented in Appendix C of the ISPF Dialog Developer’s Guide
- Trace started and stopped using TSO ISPDPTRC command



Panel Trace

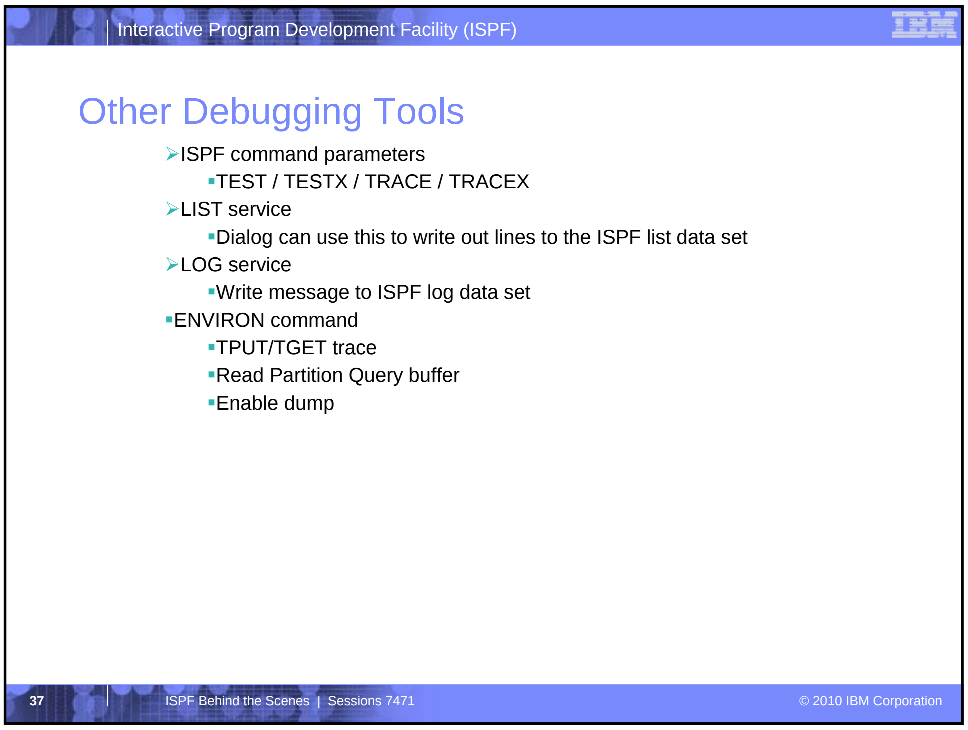
- Provides debugging capability for ISPF File Tailoring applications
- Traces the File Tailoring service calls (FTOPEN, FTINCL, FTCLOSE, and FTERASE)
- Traces ISPF processing of skeleton statements
- Trace output written to dynamically allocated variable blocked data set
- Documented in Appendix C of the ISPF Dialog Developer's Guide
- Trace started and stopped using TSO ISPFTTRC command



Dialog Test

- Dialog Test - ISPF option 7

▪Invoke dialog functions	(option 7.1)
▪Display panels and/or messages	(option 7.2)
▪List variables	(option 7.3)
▪View/modify ISPF tables	(option 7.4)
▪Browse ISPF log	(option 7.5)
▪Run ISPF services	(option 7.6)
▪Trace Dialog service calls	(option 7.7.1)
▪Trace variables	(option 7.7.2)
▪Breakpoint services	(option 7.8)

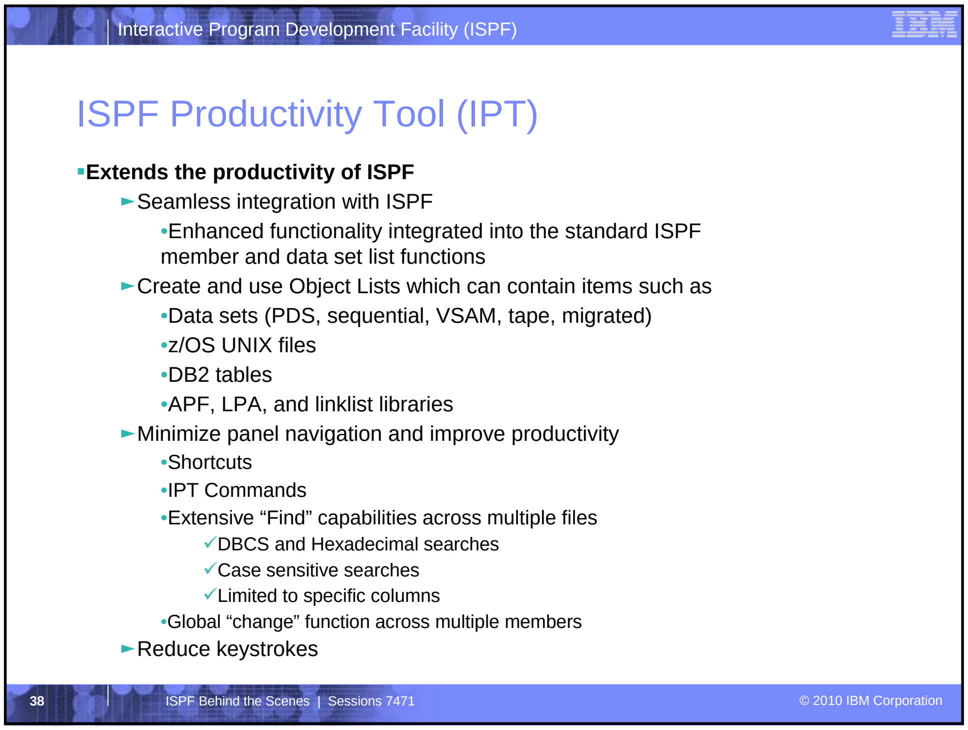


Interactive Program Development Facility (ISPF)

Other Debugging Tools

- ISPF command parameters
  - TEST / TESTX / TRACE / TRACEX
- LIST service
  - Dialog can use this to write out lines to the ISPF list data set
- LOG service
  - Write message to ISPF log data set
- ENVIRON command
  - TPUT/TGET trace
  - Read Partition Query buffer
  - Enable dump

37 ISPF Behind the Scenes | Sessions 7471 © 2010 IBM Corporation



Interactive Program Development Facility (ISPF)

ISPF Productivity Tool (IPT)

- **Extends the productivity of ISPF**
  - Seamless integration with ISPF
    - Enhanced functionality integrated into the standard ISPF member and data set list functions
  - Create and use Object Lists which can contain items such as
    - Data sets (PDS, sequential, VSAM, tape, migrated)
    - z/OS UNIX files
    - DB2 tables
    - APF, LPA, and linklist libraries
  - Minimize panel navigation and improve productivity
    - Shortcuts
    - IPT Commands
    - Extensive “Find” capabilities across multiple files
      - ✓DBCS and Hexadecimal searches
      - ✓Case sensitive searches
      - ✓Limited to specific columns
    - Global “change” function across multiple members
  - Reduce keystrokes

38 ISPF Behind the Scenes | Sessions 7471 © 2010 IBM Corporation