IBM z Systems

All about the Subsystem Interface (SSI)

Session 17117

Peter Relson, IBM
relson@us.ibm.com
Glenn Hanna, CA
Glenn.Hanna@ca.com
March 4, 2015

Abstract

- This presentation will explore the functions and attributes of the Subsystem Interface (SSI) including new function being introduced in z/OS 2.2.

What is the SSI?

- **Interface to share information**
- **You may (synchronously) request information from a subsystem**
- **You may (synchronously) send information to a subsystem**
  - **Directed to one specific subsystem**
  - **Broadcast to all (interested) subsystems**
- **The system manages the interactions – the target might not exist or might not be interested**

What is the SSI?

- A subsystem is represented by an “SSCVT” (AKA SSCT) mapped by IEFJSCVT
- CVTJESCT → JESCT. JESSSCT → first SSCVT
  - Primary JES is the first subsystem
  - MSTR is the second
- The SSCVT provides 8 bytes for user data. This was used as a way to get an “anchor”. There have been much better ways to accomplish that for a long time, now.
  - System level name/token
  - A slot in the “customer anchor table” (used by many ISV's)

What are some subsystems?

- The master subsystem MSTR
- The primary subsystem (JES2 or JES3)
- Some other IBM-defined subsystems
  - AXR
  - IRLM
  - RACF
  - SMS
- There are also other IBM-defined subsystems as well as non-IBM-defined subsystems

Defining Subsystems (IEFSSNxx)

- SSN system parameter and IEFSSNxx parmlib member
- The subsystem name is 1-4 characters. In parmlib, it must be alphanumeric or national (this is not true for IEFSSI).
- IEFSSNxx has a “positional” format. This is the “old” format. You really ought to use the “keyword” format. There is an ISPF edit macro in 'SYS1.SAMPLIB(IEFSSNXX)' to convert from old to new.
  - Positional format does not have dynamic functionality
- IEFSSNxx “keyword” format
  - Subsystem is dynamic (can be activated, deactivated)

Defining Subsystems (IEFSSNxx)

- Defaults to IEFSSN00
- Must identify a “primary” subsystem
- Defined in the provided order (except that the primary subsystem is the first SSCVT and the MSTR subsystem is the second)
  - But as of z/OS 1.12 “BeginParallel” is provided, so subsystems are not necessarily initialized in the order defined

Defining Subsystems (IEFSSNxx)

Keyword format (subsystem is considered "dynamic")

- **SUBSYS SUBNAME(subname)**
  - **[INITRTN(initrtn) [INITPARM(initparm)]]**
    - **Must be accessible via LNKLST or LPA**
  - **PRIMARY(<u>NO</u> | YES)**
  - **START(<u>YES</u> | NO)**
  - **[CONSNAME(consname)]**
    - **For initialization messages**
- **BeginParallel**
  - **Initialization routines from this point onward are invoked in parallel**

Defining/Manipulating Subsystems (SETSSI command)

- **Subsystems defined via SETSSI are considered dynamic**
- **ADD subname,CONSNAME=c,INITRTN=i,INITPARM=ip**
- **ACTIVATE subname**
- **DEACTIVATE subname**
- **DELETE subname (z/OS 2.2 only)**

Defining/Manipulating Subsystems (IEFSSI macro)

Subsystems defined via IEFSSI are considered dynamic

- **REQUEST=ADD,SUBNAME=s, CONSNAME=c,INITRTN=i,INITPARM=ip,INITPLEN=ipl**
- **REQUEST=ACTIVATE,SUBNAME=s,INTOKEN=i**
- **REQUEST=DEACTIVATE,SUBNAME=s,OUTTOKEN=o**
- **REQUEST=OPTIONS,SUBNAME=s [,COMMAND={NO|YES}] [,REQDSUB={MSTR|PRI}] [,EVENTRTN=e] <z/OS 2.2 only>**
- **REQUEST=SWAP,SUBNAME=s,INTOKEN=i,OUTTOKEN=o**
- **REQUEST=PUT,SUBNAME=s,SUBDATA=s1,SUBDATA2=s2**
- **REQUEST=GET,SUBNAME=s,SUBDATA=s1,SUBDATA=s2**

Defining/Manipulating Subsystems (IEFSSI macro)

- INTOKEN: token representing SSVT (subsystem vector table) to be used (from IEFSSVT CREATE, IEFSSI DEACTIVATE, IEFSSI SWAP)
- OUTTOKEN: output token representing SSVT for later use
- COMMAND: Does subsystem respond to SETSSI activate/deactivate? If not, error message if attempted.
- REQDSUB: For “S subname”, start under MSTR or primary subsystem
- EVENTRTN: An “exit” routine to learn of such customer-initiated events as “delete”
- SUBDATA1: get/put 1st 4 bytes of user data
- SUBDATA2: get/put 2nd 4 bytes of user data

EVENTRTN (z/OS 2.2)

- Events that it gets control for: currently, only DELETE
- Data for the events is mapped by IEFJSEPL
- Must be accessible by LNKLST / LPA at the time of the IEFSSI OPTIONS function
- EVENTRTN is provided only for IEFSSI, not for IEFSSNxx and SETSSI definitions. Regardless of how the subsystem is defined, the INITRTN can use IEFSSI to add the EVENTRTN.
- Gets control in supervisor state, key 0, event issuer's address space, primary ASC mode, P=H=S, AMODE 31, task mode, enabled for I/O and external interrupts, no locks held

EVENTRTN (cont)

- **Input regs:**
  - **0 – contains no information for use by the exit routine**
  - **1 – address of area mapped by IEFJSEPL**
  - **2 – 12 contain no information for use by the exit routine**
  - **13 – address of 72-byte savearea**
  - **14 – return address**
  - **15 – entry point address**
- **Output regs:**
  - **0-15 – not part of the interface, need not be preserved**

Defining the Subsystem Vector Table

- IEFSSVT and IEFSSVTI macros are provided to help (they supplanted IEFJSVEC when introduced 20 years ago)

- SSVT identifies for which functions the subsystem is to get control (and identifies the function routine)
  - Starts with 256 1-byte entries then 1 or more 8-byte routine entries.
  - The 1-byte entries correspond to the subsystem function code. When the entry is 0, there is no function routine and the subsystem is not interested. When the entry is non-0, it identifies which “routine entry” (the first such entry would be identified by value 1, etc.)
  - The “routine entry” may identify the name (and the system will locate this name in LPA or use LOADTOGLOBAL=YES). I strongly recommend that you use LOADTOGLOBAL=YES only if your address space can never terminate.

Defining the Subsystem Vector Table (cont)

- The “routine entry” may contain the 4-byte entry point address (in bytes 4-7 of the 8-byte entry, with bytes 0-3 hex zeroes).
- The AMODE of the function routine is determined as follows
  - When name is provided, the AMODE of the directory entry (24 or 31)
  - When address is provided, bit 0 of the address (when on, AMODE 31; otherwise AMODE 24). This bit can be set according to the FUNCAMODE keyword of IEFSSVTI

Defining the Subsystem Vector Table (IEFSSVTI)

Static definition

- IEFSSVTI TYPE=INITIAL,SSVTDATA=ssd,TABLEN=t
- (one or more) IEFSSVTI TYPE=ENTRY,
  [FUNCNAME=fn, | FUNCADDR=fa,]
  NUMFCODES=nf,
  [FCODES=(f0,...,fn)]
- IEFSSVTI TYPE=FINAL

Defining the Subsystem Vector Table (IEFSSVTI)

Dynamic manipulation

- IEFSSVTI TYPE=COPY,SSVTDATA=sd,SOURCE=ssd
- (one or more)
  IEFSSVTI TYPE=SET,SSVTDATA=sd,
  SOURCE=ssd,ENTRYDATA=n,
  [FUNCNAME=fn, | FUNCADDR=fa, [FUNCAMODE=fam,]]
  [FCODES=(f0,...,fn)]

Dynamic data definition

- IEFSSVTI TYPE=LIST (this creates a DSECT so put within your data definitions)
- IEFSSVTI TYPE=RESERVE,SSVTDATA=sd,
  {TABLEN=t | MAXFCODES=mf}

Defining the Subsystem Vector Table (cont)

-
  ```
  IEFSSVT SUBNAME=s,REQUEST=CREATE,
     SSVTDATA=sd,OUTTOKEN=o,
     SUBPOOL={s|241},
     MAXENTRIES=m,
     LOADTOGLOBAL={NO|YES},
     ERRFUNCT=e
  ```

- **SSVTDATA=sd: table defined by IEFSSVTI**

- **OUTTOKEN=o: output token representing this SSVT**

- **MAXENTRIES=m: maximum number of “routine entries”**

- **LOADTOGLOBAL: NO – routine is asserted to be in LPA; YES – use LOAD with GLOBAL=YES (see previous warning!)**

- **ERRFUNCT=e: function routine name being processed when (if) error occurred**

Defining the Subsystem Vector Table (cont)

- IEFSSVT SUBNAME=s,REQUEST=DISABLE, SSVTDATA=sd,INTOKEN=i
  - Deactivate specific function codes

- IEFSSVT SUBNAME=s,REQUEST=ENABLE, SSVTDATA=sd,INTOKEN=i,LOADTOGLOBAL={NO|YES}, ERRFUNCT=e
  - Activate (or re-activate) function codes

- IEFSSVT SUBNAME=s,REQUEST=EXCHANGE, SSVTDATA=sd,INTOKEN=i,LOADTOGLOBAL={NO|YES}, ERRFUNCT=e
  - Exchange function routines to respond to currentyl enabled function codes

Initializing the subsystem

- System LINKs to INITRTN, passing the SSCVT and a parameter list (which identifies the INITPARM)
  - R1 → 2-word area,
    - Address of SSCVT
    - Address of subsystem initialization parameter list (mapped by IEFJSIPL)

Implementing your subsystem

- Broadcast calls that your subsystem might listen for
  - (4) Late end-of-task (after many RESMGRs)
  - (8) End of memory (end of address space)
  - (9) WTO(R)
  - (10) SVC 34 (command)
  - (14) Delete Operator Message (DOM)
  - (48) Help
  - (50) Early end-of-task (before most RESMGRs)
  - (78) Tape device selection
- Directed calls that a subsystem might listen for
  - (54) Request subsystem version info
  - (58) SMF SUBPARM option change

Interacting with the subsystem (IEFSSREQ)

- IEFSSREQ (no parameters)
  - User builds “SSOB” (header: IEFSSOBH)
    - SSOBID = 'SSOB'
    - SSOBLEN = length of SSOB header
    - SSOBFUNC = function code
    - SSOBSSIB = address of SSIB or 0 (0 indicates to use the “life of job” SSIB which identifies the primary subsystem)
  - User builds “SSIB” (IEFJSSIB)
    - SSIBID = 'SSIB'
    - SSIBLEN = length of SSIB
    - SSIBSSNM = subsystem name
- User sets up R1 with address of one-word parameter list, where that word contains the address of the SSOB and has bit 0 on

Interacting with the subsystem (IEFSSREQ)

Output from IEFSSREQ

- There is an SSI return code (in reg 15)
  - SSRTOK – 0
  - SSRTNSUP – 4 - subsystem does not support this function
  - SSRTNTUP – 8 – subsystem exists but is not up
  - SSRTNOSS – 12 – subsystem does not exist
  - SSRTDIST – 16 – disastrous error
  - SSRTLERR – 20 – logical error (bad format)
  - SSRTNSSI – 24 – SSI not available
- There is a subsystem return code (in SSOBRETN)
- There may be function-dependent data returned by the subsystem

Interacting with the subsystem (IEFSSREQ)

- System SSI “router” runs in key (any), state (problem or supervisor), task/SRB mode (either), cross-memory environment (any) of invoker.
- R15 return code is “general”. SSOBRETN is the function-specific return code
- Types of call
  - Directed – target subsystem is identified in SSIBSSNM
  - Broadcast – target subsystem is MSTR. System builds a unique SSOB/SSIB for each intended subsystem, and provides back to the caller the smallest reg 15 value and the largest SSOBRETN value across all the calls

Interacting with the subsystem (IEFSSREQ)

- Requests that you might make of the primary (JES) subsystem
  - (1) Process Sysout data sets
  - (11) User Destination Validation/Conversion
  - (20) Request job ID
  - (21) Return job ID
  - (54) Request subsystem version information
  - (70) Scheduler Facilities Services
  - (71) JES Job Information
  - (75) Notify user message service
  - (79) SYSOUT Application Program Interface (SAPI)
  - (80) Extended Status Function Call
  - (82) JES properties
  - (83) JES Device information services
  - (85) JES Job Modify

Notify SSI

- Sends notification message to user
- Use SSI Function 75 (IAZSSNU Macro)
- Callers are not required to have a job structure associated with JES
- Destination can be a user on another node or member within the MAS

SYSOUT Application Program Interface SSI

- Obtains information related to SYSOUT
- Use SSI Function 79 (IAZSSS2 Macro)
- SYSOUT Selection Criteria for filtering
- Can be used with Spool Browse

Extended Status SSI

- Obtain JOB and SYSOUT Information
- Use SSI Function 80 (IAZSSST Macro)
- Information in the JES2 Checkpoint is returned
  - 3 call types
  - Get JOB data
  - Get SYSOUT and JOB data
- Release Memory
- Filters control the returned data
- Supports directed SSIs and Broadcast

JES Properties - SSI

- Sends notification message to user
- Use SSI Function 82 (IAZSSJP Macro)
- Callers are not required to have a job structure associated with JES (Directed SSI)
- Information Returned
  - NJE Nodes
  - Spool Information
  - Initiator Information
  - JESPlex Information
  - Job Class Information

JES Device Information SSI

- Sends notification message to user
- Use SSI Function 83 (IAZSSJD Macro)
- Callers are not required to have a job structure associated with JES (Directed SSI)
- Obtain information about and filter on:
  - Printers (local and remote)
  - Punches (local and remote)
  - Readers (local and remote)
  - LOGON devices
  - NETSRV devices
  - Line devices
  - Job / SYSOUT transmitters and receivers (NJE and offload)

Modify Job Function SSI

- Sends notification message to user
- Use SSI Function 85 (IAZSSJM Macro)
- Required to have a job structure associated with JES
- Allows modification of job characteristics

Interacting with the subsystem (IEFSSREQ)

- Requests that you might make of other subsystems (every subsystem ought to document the functions that it provides)
  - (15) Verify subsystem function (send to MSTR, with SSIBJBID's 1st 4 bytes identifying the subsystem) to be verified (JES does support this)
  - (54) Request subsystem version information
  - (80) Extended Status Function Call (each subsystem may define the data it supports and behavior that it provides for this function)

IEFSSI QUERY

Extract data about one or more subsystems

```
IEFSSI REQUEST=QUERY,SUBNAME=s,
   WORKAREA=w,WORKASP=wsp
```

- Subsystem name may be wildcarded. Info is returned for all matching names (e.g., active or inactive, does it respond to commands, what are the function codes)
- Workarea is mapped by IEFJSQRY
- WORKASP identifies the subpool to use (the system obtains the storage; the user is responsible for freeing the storage)

SSI DELETE (z/OS 2.2)

The problems

- If the INITRTN has a basic problem (such as “does not exist”) it is not possible to “re-do”
- If a subsystem is installed, there is no way to change its init parameters and start over

The solutions

- Do some preliminary checking of INITRTN so that on some normal problems the subsystem is not even defined
- Provide a logical deletion function

SSI DELETE (cont)

INITRTN problem detection

- **If the LOAD fails (name is wrong, or name is right but is not in the LNKLST, or not in an APF-authorized data set), the subsystem define is rejected:**

  ```
  IEFJ027I SUBSYSTEM INITIALIZATION ROUTINE
  initialization-routine NOT FOUND FOR
  SUBSYSTEM ssname
  ```
- **This occurs for all subsystem defines (whether by IEFSSNxx parmlib member, SETSSI command, IEFSSI macro)**

SSI DELETE (cont)

Logical Deletion

- Does not free storage related to the subsystem
- Does not terminate subsystem routines currently in control
- Does stop making new calls to subsystem routines
- Does remove from the SSCVT chain

SETSSI DELETE,SUBNAME=s,FORCE

- Subsystem does not need to be dynamic
- Use at your own risk (especially if you're going to try again, as perhaps the initrtn “did something” that will not play well with a second try)
- EVENTRTN is driven if subsystem is dynamic
- Special SSCVT entries are created (SSCTSNAM has !DEL or !DMY)

SSI etc

- Some subsystems support the concept of a “subsystem data set”
- IEFJFRQ installation exit
- Subsystem affinity service (SSAFF) – largely supplanted by task-level name/token

Summary

- The subsystem interface provides mechanisms to communicate with the primary subsystem (and other subsystems) and also to interact with certain system events.
- With a dynamic subsystem, you can change between having the subsystem be active and inactive.
- With z/OS 2.2, you can address some InitRtn errors and “try again” and can get rid of a subsystem that was temporarily added such as for test purposes.

References

Publications

- z/OS V2R1 MVS Authorized Assembler Services Reference
- z/OS V2R1 MVS Initialization and Tuning Reference
- z/OS V2R1 MVS System Commands
- z/OS V2R1 MVS Using the Subsystem Interface

Questions?