JES2 Bootcamp - Part 3 of 3
Customization and other considerations

Tom Wasik
IBM Rochester, MN
wasik@us.ibm.com

Thursday 9:30 AM
Session Number 14287

Trademarks

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries.

- IBM®
- MVS™
- Redbooks®
- RETAIN®
- z/OS®
- zSeries®

The following are trademarks or registered trademarks of other companies.

- Java and all Java-related trademarks and logos are trademarks of Sun Microsystems, Inc., in the United States and other countries.
- Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
- Microsoft, Windows and Windows NT are registered trademarks of Microsoft Corporation.
- UNIX is a registered trademark of The Open Group in the United States and other countries.
- SET and Secure Electronic Transaction are trademarks owned by SET Secure Electronic Transaction LLC.
- All other products may be trademarks or registered trademarks of their respective companies.

Notes:

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM Business contact for information on the product or services available in your area.

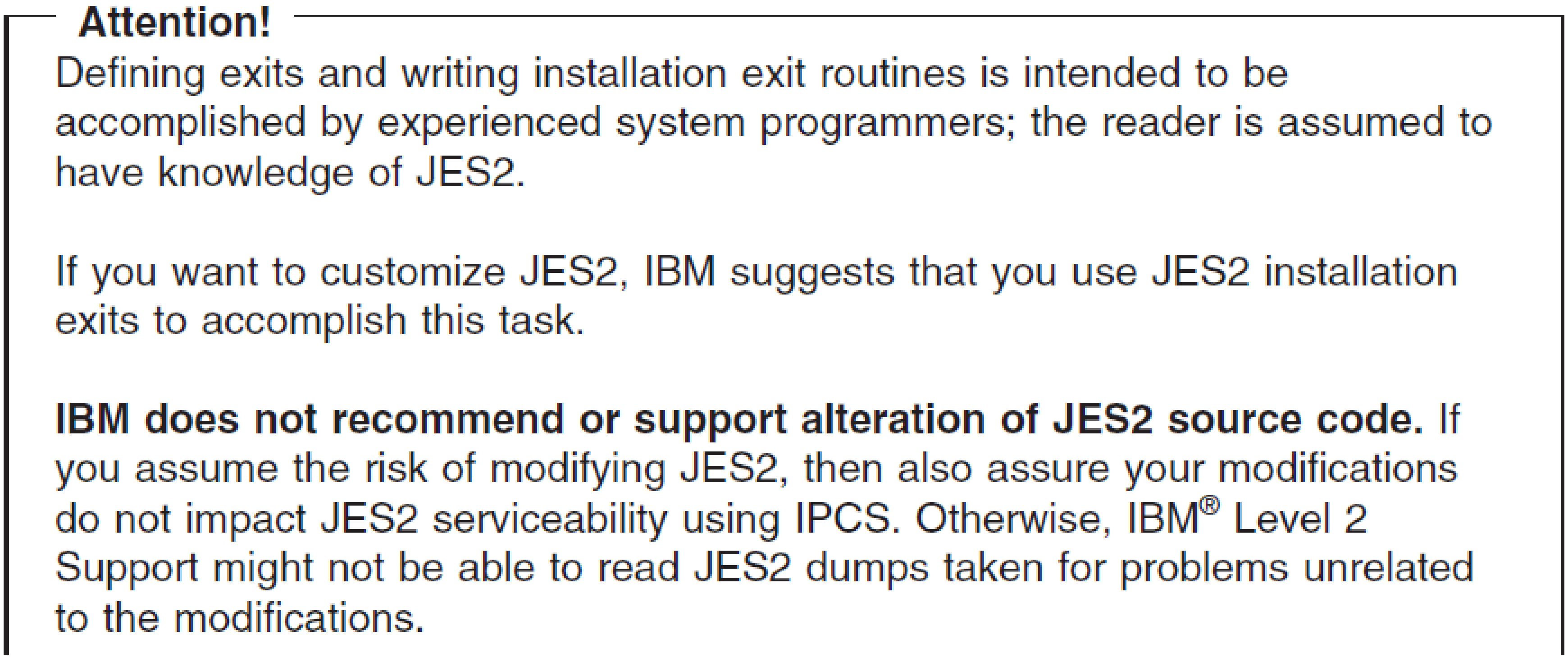
All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

JES2 Exits

- From JES2 exits chapter 1 page 1



Attention!

Defining exits and writing installation exit routines is intended to be accomplished by experienced system programmers; the reader is assumed to have knowledge of JES2.

If you want to customize JES2, IBM suggests that you use JES2 installation exits to accomplish this task.

IBM does not recommend or support alteration of JES2 source code. If you assume the risk of modifying JES2, then also assure your modifications do not impact JES2 serviceability using IPCS. Otherwise, IBM® Level 2 Support might not be able to read JES2 dumps taken for problems unrelated to the modifications.

JES2 Exits

- In JES2 exits are considered extensions to the product
  - This historically is what customers wanted
  - Give exit control at a point in processing and let it do what it wants
  - No limits on what an exit can do
- This, combined with availability of source leads to
  - Exits that can do amazing functions (eg what some vendors do)
  - Exits that are very sensitive to JES2 internal data structures
  - Exits that need to be updated every release to continue working
- Because of this, JES2 requires JES2 exits be re-assembled for every release
  - When fields change in nature, they are renamed to trigger assembly errors in exits
  - Assembly errors are GOOD because they tell you a change is needed
- Having said this, reality is
  - Most exits recompile and work with the new release
  - JES2 development focuses on NOT breaking exits
  - Newer exits have formal parameters to reduce dependence on internal structures

JES2 Exits

- Each exit point in JES2 has an exit number
- Exit code must be is placed in a JES2 format module
  - Has a $MIT and $MITABLE
  - Loaded by the JES2 LOADMOD statement
- An exit routine must be identified with a $ENTRY macro
- Each exit number can have multiple exit routines associated with it
  - Specify on the ROUTINES= keyword of the EXIT statement
  - Called in the order listed
- Exit return code are defined as
  - 0 – exit successful, continue calling routines
  - 4 – exit successful, stop calling exit routines
  - >4 – exit failed, stop calling exit routines

JES2 Exits - Environments

- All code in JES2 has an associated assembly environment
- Environment defines what services/macros are available
    - Some macro expand differently base on environment
- Environment also defines register contents
- Exit environments
    - JES2 - JES2 main task
    - SUBTASK - JES2 subtask
    - USER - JES2 USER environment
    - FSS - Functional subsystem environment
- Set on $MODULE ENVIRON= or $ENVIRON macro
- If you get this wrong, EXIT will probably ABEND
    - Wild branch is a typical problem

JES2 Exits - Environments

- JES2 Main Task environment– ENVIRON=JES2
- JES2 address space, JES2 Main task
  - MVS WAITs vs JES2 $WAIT is exit dependent
  - R11 = HCT address
  - R13 = PCE address
- JES2 Main Task Serialized
  - Only one PCE runs at a time
- Routine should reside in private storage
  - LOAD(routine) STORAGE=PVT
- Routine may reside in common storage
  - Why use up CSA unnecessarily?

JES2 Exits - Environments

- Avoid MVS WAITs in JES2 environment!
  - Use $WAIT instead
    - Caution, some exits do not even allow $WAIT!
  - Use JES2 equivalent services that do not MVS WAIT
    - $WTO, etc.
  - If you must call a service which can MVS WAIT
    - Use the $SUBIT service to send request to a subtask
      - *PCE $WAITs while subtask does the MVS WAIT.*
  - Exceptions - JES2 Initialization/Termination (Exits 0, 19, 24, 26)
    - JES2 dispatcher not enabled, so CAN NOT $WAIT
    - ENVIRON=(JES2,INIT) can be used in 0, 19, 24
    - ENVIRON=(JES2,TERM) can be used in 26

JES2 Exits - Environments

- JES2 Subtask environment – ENVIRON=SUBTASK
- JES2 address space, non-main task
  - MVS WAIT can be done
  - $WAIT not allowed
  - R11 = HCT
  - R13 = DTE
- Multi-tasking considerations apply!
  - Multiple subtasks can simultaneously be in subtask
  - Reentrancy is very important
- Routine should reside in private storage
- Routine may reside in common storage

JES2 Exits - Environments

- JES2 USER environment – ENVIRON=USER
- Any address space, any task
  - R11 = HCCT
  - R13 = Available save area
- Multi-tasking considerations:
  - Multiple tasks in multiple address spaces may be in exit simultaneously
  - Reentrancy very important
- Routine MUST reside in common storage
  - LOAD(routine) STORAGE=CSA
  - LOAD(routine) STORAGE=LPA
- Exits may get control when JES2 address space is down!

JES2 Exits - Environments

- JES2 USER Environment – ENVIRON=(USER,ANY)
- Any address space, any task
  - R11 = HCCT
  - R13 = Available save area or PCE
- Same as ENVIRON=USER except:
  - $SAVE/$RETURN services based on caller
    - PCE caller use JES2 main task logic
      - *No BAKR's (so $WAIT can be done if main task)*
    - Not PCE use USER environment logic
- Useful for routines that could be called in the user environment OR under main task
  - Detect run-time environment by checking
    - PSAAOLD, PSATOLD for JES2 Main task
    - R13 eyecatcher for ‘PCE’

JES2 Exits - Environments

- Functional subsystem environment – ENVIRON=FSS
- FSS address space
  - R11 = HFCT
  - R13 = Save area
- Routine MUST reside in common storage

JES2 Exits – basic macros

- The structure of an exit (without code or comments)

```
         COPY   $HASPGBL
         $MODULE ENVIRON=xxxx

ROUTINE1 $ENTRY SAVE=YES
         $ESTAE <- not required but recommended
... Insert code here
         $RETURN

         $MODEND
```

JES2 Exits – $XPL

- Many exit (all newer ones) pass an XPL
  - Generally in register 1 but older exits may be in R0 or R2
- XPL has a header to describe the exit being called

XPLID	Eye catcher
XPLLEVEL	Version number for base section
XPLXITID	Exit id number
XPLEXLEV	Version number for specific exit XnnnVERN is the equate)
XPLIND	Indicator byte – Field for why called
XPLCOND	Condition byte – Bits for conditions
XPLRESP	Response byte - Bits for exit response (Modifiable by Exit routine)
XPLSIZE	Size of parameter list including the base section

JES2 Exits – $XPL

- Header followed by exit specific fields
    - Label format is XnnnDATA where nnn is the exit number
        - Eg X001JCT
- Equate over XPLIND, XPLCOND, and XPLRESP
    - Eg X001IND, X001COND and X001RESP
- Note that XPLRESP may start off with bits sets
- XPL is more formal way to interface to JES2
    - When possible set XPL field instead of control block field
- JES2 Exits manual describes interface to each exit
    - This includes fields defined in the XPL

JES2 Exits – Control blocks

- JES2 data areas can be expanded by exits
  - Some data areas have user fields built in
    - $USER1-5 in HCT data area
    - CCTUSER1-4 in HCCT data area
    - SJBUSER in SJB data area
    - JCTUSER1-F in JCT data area
  - Other can be extended using services
    - $JCTXxxx service to extend the SPOOLed JCT data area
    - $BERTTAB tables to extend certain checkpointed data areas
  - Some services support use of named tokens
    - $SCANTABs, $PCETAB, etc

JES2 Exits – Control blocks

- Some control blocks are intended just for exit use
  - UCT – pointed by $UCT field in HCT
  - CUCT – pointed by CCTCUCT field in HCCT
  - UPADDR – pointed to by $UPADDR field in HCT
  - DUCT – pointed to by name/token pair (HOME level)
  - DCCT – pointed to by name/token pair (SUBSYS level)
  - Etc (see JES2 Exits manual for more information)
- User related data areas can be defined in $USERCBS
  - Macro supported by $MODULE to define DSECTs

JES2 Exits – Table Pairs

- Not all JES2 customizations are done in exits
- Table pairs extend/customize JES2 processing
  - Pair is a misnomer – Support allows more than 2 tables
    - JES2 and User table (origin of pair concept)
    - PLUS dynamic tables – unlimited number of additional tables
- Table pairs can be used to
  - Add installation commands or keywords to command
  - Override default processing for a command keyword
  - Replace the text in messages issued by $BLDMSG
  - Add structures like PCEs, DTEs, PC routines, etc
  - Extend data area via $BERTTABs

JES2 Exits – Table Pairs

- The MCT is the home for tables that have pairs
- There are 2 major classes of table
  - Commands and their operands ($SCANTABs)
  - Other structures supported by $GETABLE
    - PCE TABLE ($PCETAB)
    - DCT TABLE ($DCTTAB)
    - DTE TABLE ($DTETAB)
    - TRACE ID TABLE ($TIDTAB)
    - PC Routine table ($PCTAB)
    - BERT table ($BERTTAB)
    - Work selection ($WSTAB)
- These can be customized to some extent by installations

JES2 Exits – Table Pairs - $SCANTAB

- Probably the most common to modify
  - Used for commands, initialization statements, messages
- Commands and init statement syntax is in levels

```
$T CKPTDEF,CKPT1=(DSN=SYS1.HASPCKPT)
```

- Components of this command are
  - Verb – $T
  - Object – CKPTDEF
  - Keyword level 1 – CKPT1
  - Keyword level 2 – DSN
- The verb is not controlled by table pairs
- The object and keywords are controlled by tables

JES2 Exits – Table Pairs - $SCANTAB

- Extend or modify a $SCANTAB table use table name
  - Main operand table (CKPTDEF level) – MCTMPSTP
    - Just have to know this one
  - First level keywords (CKPT1 level) – MCTCKTTP
    - SCANTAB= value from CKPTDEF $SCANTAB
  - Second level keywords (DSN level) – MCTKPNTP
    - SCANTAB= value from CKPT1 $SCANTAB
- Add or modify using table name in a table of $SCANTABs

```
USERCKPT $SCANTAB TABLE=(DYNAMIC,MCTCKTTP)
        $SCANTAB NAME=MAXCKPT_4K,CB=HCT,
         CBIND=($CKW,HCT,L),DSECT=CKW,
         FIELD=CKWMAXRC,RANGE=(1,X'FFFFFFFF'),CONV=NUM,
         CALLERS=($SCDCMDS)
        $SCANTAB TABLE=END
```

- This adds a MAXCKPT= keyword to $D CKPTDEF
  - Displays size of the CKPT when all keywords are set to max
    - JQENUM, JOENUM, etc.

JES2 Exits – Table Pairs - $SCANTAB

- To use this extension, add it to a module and load it
- Basic structure is similar to a basic exit

```
 COPY  $HASPGBL
$MODULE ENVIRON=xxxx


$SCANTAB …


$MODEND
```

- The writing of the $SCANTAB is beyond this presentation
- See http://www.share.org/d/do/7211 for more details

JES2 Exits – Table Pairs - $SCANTAB

- Example from HASX00A sample

```
MAINPARM $SCANTAB TABLE=(DYNAMIC,MCTMPSTP)
         $SCANTAB NAME=TESTDEF,CONV=SUBSCAN,MINLEN=4,
               CMDRDIR=DEF,SCANTAB=(X0TBTSTP,ADDR),MSGID=949,
               CALLERS=($SCIRPL,$SCIRPLC)  INIT STMT
        $SCANTAB TABLE=END
X0TBTSTP $PAIR ,
        $SCANTAB NAME=IRPLERRS,FIELD=CIRFLAG2,CB=PCE,CONV=FLAG,
               VALUE=(YES,0,FF-CIRF2SSE,NO,CIRF2SSE,FF),MINLEN=4,
               CALLERS=($SCIRPL,$SCIRPLC)
        $SCANTAB TABLE=END
```

- Creates new initialization statement TESTDEF
- Creates new keyword on statement IRPLERRS=YES|NO
  - If IRPLERRS=NO, then set a bit CIRF2SSE
  - Bit suppresses WTOR when an initialization error is detected
- Who said sample exits are not useful?

JES2 Exits – Table Pairs – Other tables

- Some messages in JES2 are generated using $SCANTAB
- These also have a table pair for the FULL message
  - Not for keywords or pieces of messages
- This messages are build using the $BLDMSG service
- Can be used to issue new message or replace JES2's
  - Table name is MCTMGTP
- Caution when replacing JES2 processing
  - Can be confusing or could miss service update
- Basic technique
  - Start with JES2 text from HASPMSG or HASCBLDM
  - Copy to your load module and modify $SCANTABs
  - Assemble and load updated message

JES2 Exits – Table Pairs – Other tables

- PCEs/DTEs are another good candidate
  - Can create your own PCE to perform functions
  - Can add a new subtask for special processing
- No need to know table name for these

```
$PCETAB TABLE=DYNAMIC
$DTETAB TABLE=DYNAMIC
```

- Do not recommend overriding JES2 tables
  - Though possible to do
- Need code to actually start the PCE or SUBTASK
  - $PCEDYN and $DTEDYN ATTACH macros should be used
  - Can be placed in $$$$LOAD routines

JES2 Exits – Load and delete routines

- Some load modules need initialization routines
  - Need to set up storage, start PCEs/DTEs, etc
- One solution is to use an initialization exit to do function
  - Exit 24 often used, but can also use exit 0 or 19
- Routine $$$$LOAD get called when module is loaded
  - Must be defined with a $ENTRY
  - Must be in first CSECT in the load module
- Routine runs in the JES2 main task after module is loaded
  - Better because it is called for dynamic load of module
    - $ADD LOAD command

JES2 Exits – Load and delete routines

- Similar to initialization, some modules need cleanup
  - Delete data areas or stop processes that they run
- Again, can be done in termination exit
  - Exit 26 is standard for this
- Routine $$$$DEL get called when module is deleted
  - Must be defined with a $ENTRY
  - Must be in first CSECT in the load module
- Routine runs in the JES2 main task as part of delete
  - Can prevent or delay module deletion
  - Better than exit since it is called for dynamic delete of module
    - $DEL LOAD command

JES2 Exits – Dynamic exits

- The LOADMOD initialization statements loads modules
  - Special case processing exists for exit 0
- The $$$$LOAD routine is called as part of loading module
- Modules can be refreshed by the $T LOADMOD command
  - Assuming they support this function
  - $$$$LOAD from new module is called
- Alternatively, exits can be loaded by $ADD LOADMOD
  - This dynamically brings in a new load module
  - Can be used for temporary functions
    - One time cleanup or setup processing
    - Diagnostic routines
    - Trying new exits in a test LPAR

JES2 Exits – Dynamic exits

- At JES2 termination exits are deleted
  - CSA/LPA exits are explicitly deleted and $$$$DEL called
  - Private exits do not call $$$$DEL routines
- Are also deleted as part of refresh processing
  - Assuming the module supports refresh
  - $$$$DEL from old module called as part of delete
- Can also delete modules that are no longer needed
  - $DEL LOADMOD command deleted module
  - Logically disconnected and eventually deleted
  - $$$$DEL called as part of process

JES2 Exits – Dynamic exits

- Not all exits can be dynamic
  - Too dependent on data areas built at initialization
  - Function too critical to run without
- Dynamic function can be disabled on $MODULE
  - DYNAMIC=NO keyword prevents dynamic processing
- Other considerations exist
  - See http://www.share.org/d/do/5781 for more information

JES2 Exits

- There is a wealth of resources on JES2 exits
  - IBM publication like
    - JES2 Exits manual
    - JES2 Macros
    - JES2 Data areas
  - Sample exits shipped in SYS1.SHASSAMP
  - SHARE presentation
  - IBM education assistant

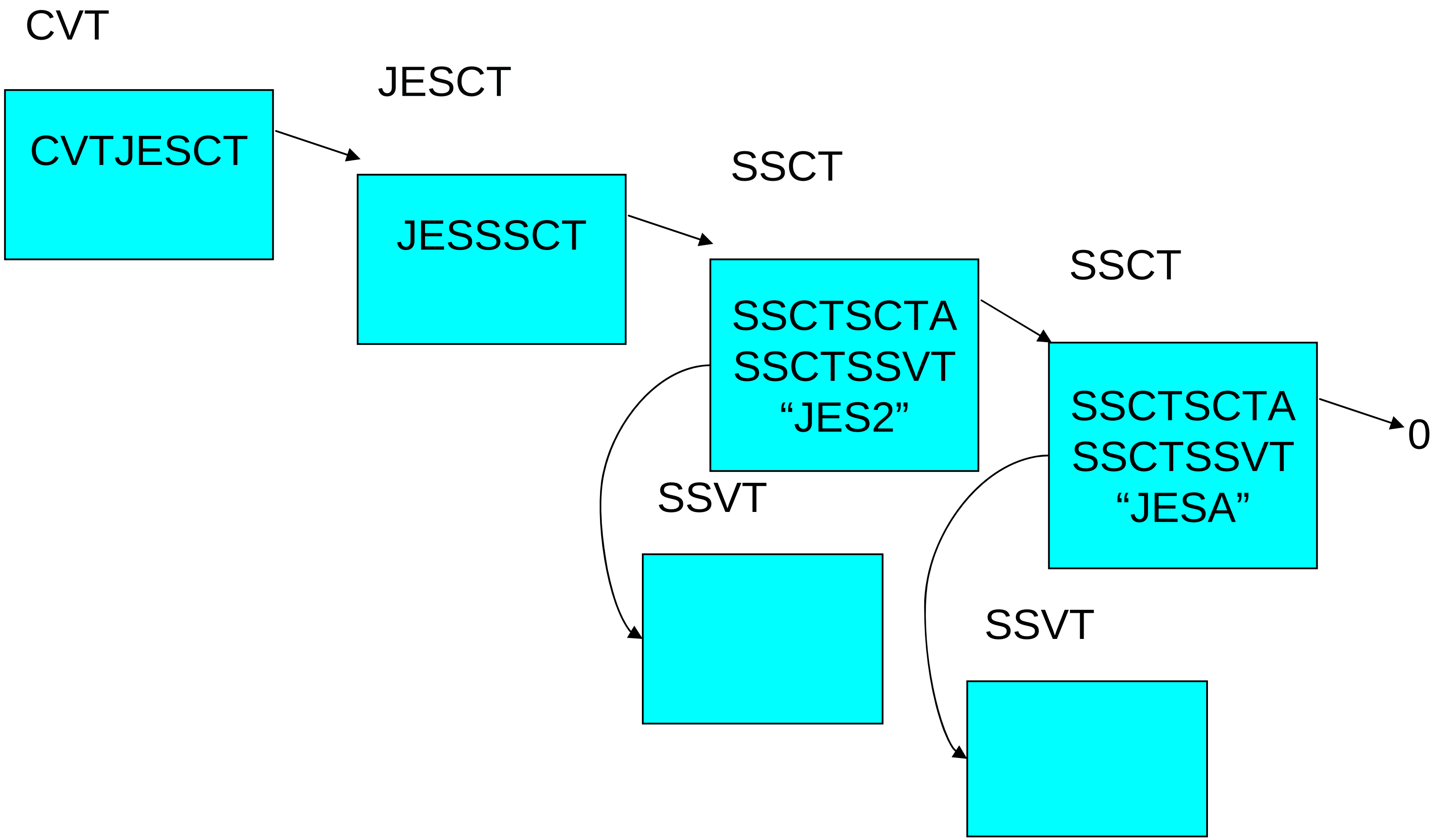
http://publib.boulder.ibm.com/infocenter/ieduasst/stgv1r0/index.jsp
http://publib.boulder.ibm.com/infocenter/ieduasst/stgv1r0/topic/com.ibm.iea.zos/zos/1.7/JobEntrySS/JES2_Exits_Overview.pdf

  - Google searches on JES2 exits
    - Most references were found on google

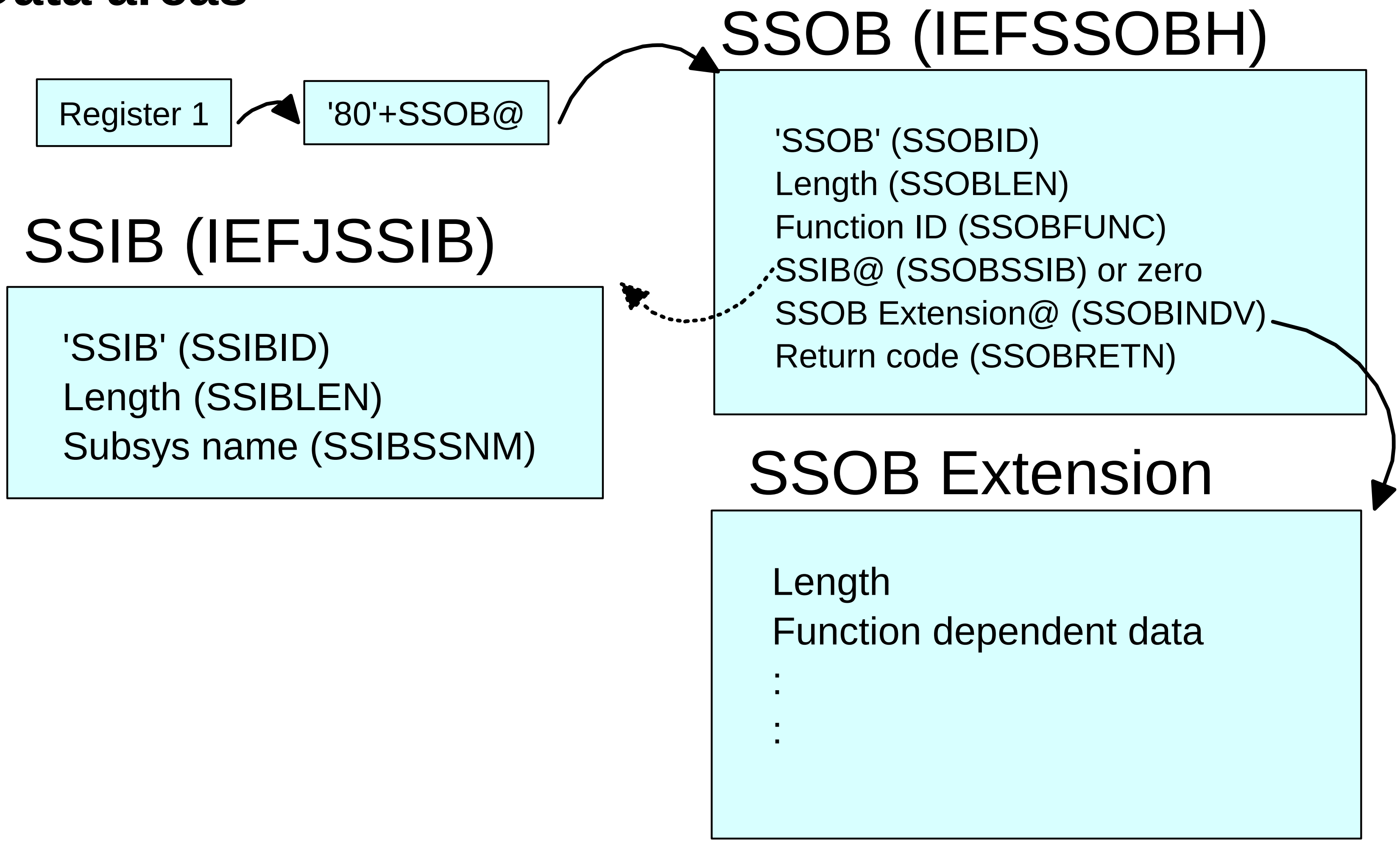
What is the SSI

- The SSI is an MVS interface to "Subsystems"
  - Used as a hook to give info to subsystems
  - WTO, CMDs, EOT, EOM, etc.
  - Used as a way to request functions
    - PSO, SAPI, Extended Status
  - Each SSI has a number and an SSOB extension
  - Subsystem identifies what it supports
  - Caller can specify subsys to process request
    - Default, Specific, All

Basic data structure



Invoking the SSI - Data areas



Register 1

'80'+SSOB@

SSOB (IEFSSOBH)

'SSOB' (SSOBID)
Length (SSOBLEN)
Function ID (SSOBFUNC)
SSIB@ (SSOBSSIB) or zero
SSOB Extension@ (SSOBINDV)
Return code (SSOBRETN)

SSIB (IEFJSSIB)

'SSIB' (SSIBID)
Length (SSIBLEN)
Subsys name (SSIBSSNM)

SSOB Extension

Length
Function dependent data
:
:

Invoking the SSI - Code

```
         USING SSOB,MYSSOB          Establish SSOB addressability
         SPACE 1
         XC    MYSSOB,MYSSOB        Zero SSOB area
         LA    R6,MYSSOB            Get address of SSOB
         SPACE 1
         MVC   SSOBID,=C'SSOB'      Set SSOB eyecatcher
         MVC   SSOBLEN,=Y(SSOBHSIZ)  Set length of SSOB header
         MVC   SSOBFUNC,=Y(SSOBSSxx)  Set function code
         MVC   SSOBSSIB,=F'0'       Use LOJ SSIB
         LA    R0,SSOB+SSOBHSIZ     Point to SSOB extension
         ST    R0,SSOBINDV          Point base to extension
         SPACE 1
         USING SSxxxxx,SSOB+SSOBHSIZ  SSOB extension addr'blty
         SPACE 1
* Code to set up SSOB extension goes here
         SPACE 1
         LA    R6,MYSSOB            Point to SSOB
         O     R6,=X'80000000'      Set HI BIT to indicate last
         ST    R6,PARMPTR           Save SSOB address in parm
         LA    R1,PARMPTR           Get pointer to SSOB
         SPACE 1
         IEFSSREQ                   Invoke the SSI
         SPACE 1
         LTR   R15,R15              If this is nonzero
         JNZ   SSREQERR              we're in big trouble
         CLC   SSOBRETN,=A(0)         Is there an error?
         JH    SSOBERR                  Yes, process error
```

What is the SSI (cont...)

The SSI calls (that applications can use) which JES supports are:

Number	Symbol	Macro	Description
1	SSOBSOUT	IEFSSSO	Process SYSOUT
2	SSOBCANC	IEFSSCS	Job cancel
3	SSOBSTAT	IEFSSCS	Job status
11	SSOBUSER	IEFSSUS	Destination validation
20	SSOBRQST	IEFSSRR	Request job ID
21	SSOBRTRN	IEFSSRR	Return job ID
54	SSOBSSVI	IEFSSVI	Subsystem information
70	SSOBSFS	IAZSSSF	SJF spool services
71	SSOBSSJI	IAZSSJI	Job information (Much of this SSI has been depricated)
75	SSOBSSNU	IAZSSNU	User notification
79	SSOBSOU2	IAZSSS2	SYSOUT API (SAPI)
80	SSOBESTA	IAZSSST	Enhanced status information
82	SSOBSSJP	IAZSSJP	JES properties
83	SSOBSSJD	IAZSSJD	JES device information
85	SSOBSSJM	IAZSSJM	Job modify

System z Social Media

- System z official Twitter handle:
  - @ibm_system_z
- Top Facebook pages related to System z:
  - Systemz Mainframe
  - IBM System z on Campus
  - IBM Mainframe Professionals
  - Millennial Mainframer
- Top LinkedIn Groups related to System z:
  - Mainframe Experts Network
  - Mainframe
  - IBM Mainframe
  - System z Advocates
  - Cloud Mainframe Computing
- YouTube
  - IBM System z

- Leading Blogs related to System z:
  - Evangelizing Mainframe (Destination z blog)
  - Mainframe Performance Topics
  - Common Sense
  - Enterprise Class Innovation: System z perspectives
  - Mainframe
  - MainframeZone
  - Smarter Computing Blog
  - Millennial Mainframer

Questions?

Questions?

Session 14287