JES2 z/OS 1.7
Migration Considerations

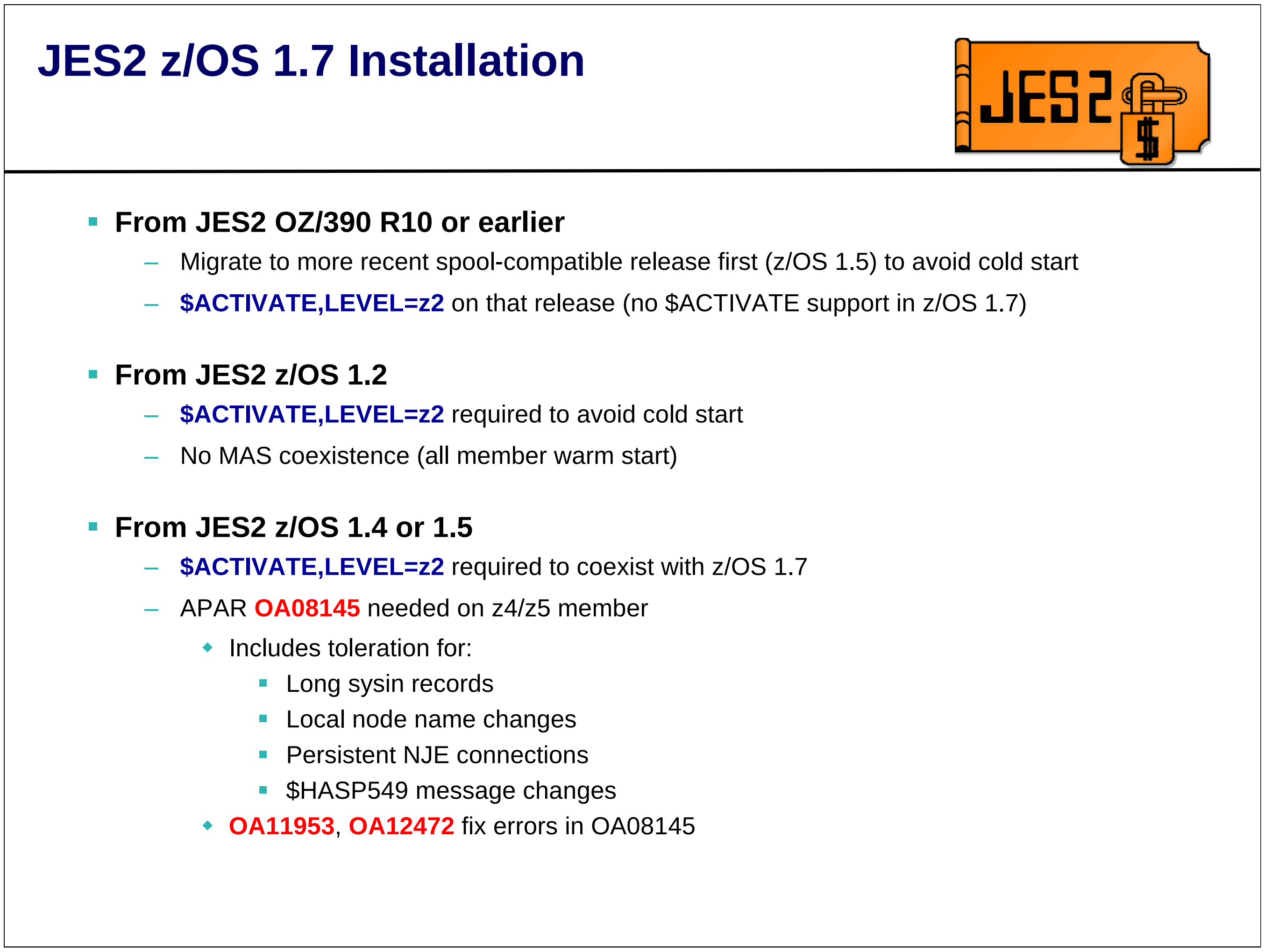
SHARE 105, Session 2656
Wednesday, August 24, 2005

Chip Wood
JES2 Design/Development/Service
Poughkeepsie, NY
chipwood@us.ibm.com

JES2 z/OS 1.7

- **New changes in z/OS 1.7**
  - NJE over TCP/IP
  - Large (>64K track) SPOOL Data Sets
  - Reader/NJE exit changes
  - Long SYSIN support (32K LRECL)
  - Table pair enhancements
  - $SCAN from non-main task environments
  - SSI for JES2 monitor information
  - SAPI and extended status enhancements
  - Checkpoint recovery (DAS corruption)
  - Other goodies (requirements!!!!)
  - Drop of support for release 4 mode

JES2 z/OS 1.7 is the largest and most function-rich release of the product in many years. This session will provide an overview of the function added in the release. Many of the functions will be discussed in much more detail in sessions throughout the week.



JES2 z/OS 1.7 Installation

- **From JES2 OZ/390 R10 or earlier**
  - Migrate to more recent spool-compatible release first (z/OS 1.5) to avoid cold start
  - **$ACTIVATE,LEVEL=z2** on that release (no $ACTIVATE support in z/OS 1.7)
- **From JES2 z/OS 1.2**
  - **$ACTIVATE,LEVEL=z2** required to avoid cold start
  - No MAS coexistence (all member warm start)
- **From JES2 z/OS 1.4 or 1.5**
  - **$ACTIVATE,LEVEL=z2** required to coexist with z/OS 1.7
  - APAR **OA08145** needed on z4/z5 member
    - Includes toleration for:
      - Long sysin records
      - Local node name changes
      - Persistent NJE connections
      - $HASP549 message changes
    - **OA11953**, **OA12472** fix errors in OA08145

To migrate to JES2 z/OS 1.7 via warm start, you must be $ACTIVATEd at the z2 level.

JES2 z/OS 1.7 Installation

- **New JES2 distribution library**
  - Moved from **SYS1.SHASLINK** (PDS) to **SYS1.SHASLNKE** (PDSE)
    - Allows HASJES20 to be a split load module
    - Most JES2 modules are now loaded above 16M
      - 1.2M of code was moved
      - A few modules had to stay below 16M
        - Some code ($SAVE, $RETURN, etc) has to tolerate AMODE 24 entry
        - Appendages, etc. that must reside below 16M (HASPBSC, HASPPRPU)
        - Extensive use of AL3 for addresses (HASPSNA, HASPCOMM)
  - **Check your JES2 proc!!!!!!**
- **If you do your own assembly/linkedit of JES2**
  - JES2 modules should be assembled with the **XOBJECT** and **LIST(133)** parameters
  - JES2 should be linkedited into a PDSE with the **RMODE(SPLIT)** option
- **CDE structure for HASJES20 is different because of this change**
  - Code that uses CDEs to find HCT may need to be changed

A significant change to the packaging of JES2 – the load library is now a PDSE. This was done to take advantage of the RMODE(SPLIT) binder option to load most of the HASJES20 load module above the 16M line. The new library is **SYS1.SHASLNKE.**

Drop of Support for Release 4 mode

- **JES2 currently support 2 modes of checkpoint operation**
  - R4 mode to be compatible with pre z/OS 1.2 members
  - z2 mode to support new features added in z/OS 1.2
- **$ACTIVATE is used to switch between 2 modes**
- **“N-3” release is z/OS 1.2**
  - All compatible releases support z2 mode
    - Note: z/OS 1.2 is no longer supported
- **JES2 z/OS 1.7 requires MAS to be in z2 mode to migrate to or coexist**
- **$ACTIVATE command deleted in z/OS 1.7 (again)**
  - Don’t worry, it will be back!
- **Action: Ensure your exits will work in z2 mode**
- **Action: Issue $ACTIVATE,LEVEL=z2 prior to migration**

Changes from R4 mode to z2 mode

- **Job number fields have changed from 2 bytes to 4 bytes**
  - Most have been renamed
  - JQE has been remapped to accommodate 4-byte fields
  - "_R4" compatibility fields have been deleted in z7
- **JQE and JOE chaining fields change from offsets to indices**
  - "_R4" compatibility fields have been deleted in z7
  - Can use $#INO, $#OIN, $QINO, #QOIN "helper macros" to convert offsets to indices and vice versa
  - Use $#JOE and $QJQE to run chains as long-term solutions
- **Job id fields can now include up to 7 digit number**
  - ***Jnnnnnnn***, ***Snnnnnnn***, ***Tnnnnnnn*** instead of **JOB*nnnnn***, **STC*nnnnn***, **TSU*nnnnn***
  - Job ids are found in many places
    - JES2 control blocks
    - JMR, JSAB
    - WQE (and therefore messages in job logs, SYSLOG, MPF exits, etc)
    - SMF records
    - SAF profiles for spool data sets
- **See also JES2 z/OS 1.2 Migration Considerations – session 2656 from SHARE 99 (Summer 2002, San Francisco)**

There are a number of things that need to be checked before your exits can run in z2 mode. If you haven't updated your exits to run in z2 mode yet, you must do so before migrating to z/OS 1.7. This presentation will not discuss those specific changes in great detail – these changes can fill a complete presentation by themselves. Those changes have been discussed in the past at SHARE, and the z/OS 1.2 Migration presentation has a lot of good reference material in it.

Large spool dataset support (>64K tracks)

- **SPOOL addressing uses 4 byte MTTR**
  - M is SPOOL extent number
  - TT is track address (up to 64K)
  - R is record number
- **SPOOL address still 4 bytes in most places**
- **4 bits from R given to TT**
  - MTTtr gives 20 bits for track address or 1M tracks
- **Limited to R values of 15 or less**
  - Buffer sizes of **2944** or greater on a 3390
- **Some data areas use 6 byte MQTR (MTTTTR) and 5 byte MQTs**

- **Use SPOOL read SSI (subfunction of SSI 71) rather than directly converting MTTRs to real track addresses**

JES2 uses 4 byte MTTRs to address records on SPOOL. Using this scheme, we can address up to 64K tracks with 255 records per track. But JES2 formats the tracks with much less than 255 records per track. On a 3390 with the recommended buffer size of 3992 bytes, JES2 used 12 records per track. This implies that we can use some of the bits from the "R" value to supplement the TT value. By borrowing 4 bits, we can get 20 bits or 1M tracks. The problem is, if the buffer size is too small, such that there are more than 15 records per track, this scheme cannot be used. That is considered a permanent restriction of this support.

So in addition to changing the MTTR, we also needed to save these 20 bit track values in TGAEs in IOTs. The problem is that a TGAE in an IOT is only 3 bytes long. To solve that problem, a new 5 byte MTTTT was defined (called MQT for short, M quad T) to use where the 3 byte MTTs used to be used. In addition, there are places in the code that use a 6 byte MTTTTR (or MQTR for short).

The SPOOL read SSI makes these changes transparent to any application that does not look at the contents of the MTTR.

Commands and Init Statements

- **Support activated by SPOOLDEF LARGEDS=**
  - **LARGEDS=FAIL**
    - All volumes must be less than 64K tracks
    - Compatible with older releases of JES2
    - MTTtrs and MQTs never used
  - **LARGEDS=ALLOWED**
    - IOT, TGAEs and other data areas converted to MQTs
    - Large volumes can be started
    - MTTtrs used for large volumes only
    - Older release can NEVER be warm started again
  - **LARGEDS=ALWAYS**
    - Same as ALLOWED but MTTtrs are used for all new volumes

Because of the nature of the changes, down level releases cannot support these new larger SPOOL volumes (in part because DFSMS on the down level MVS do not support the larger data sets). As a result, a new external was needed to “activate” the support. The LARGEDS= parameter was added to SPOOLDEF for this line item. LARGEDS has 3 values, FAIL, ALLOWED and ALWAYS. FAIL does not allow any large data set to be used. It also allows down level JES2 member to co-exist with this level of JES2. ALLOWED will activate the LARGEDS support (update checkpoint and change the format for new SPOOL control blocks) and allow large data set to be started. ALWAYS is similar to ALLOWED except you do not need a large volume to test the new format for MTTR. As such, ALWAYS is intended as a testing tool.

Once LARGEDS is set to ALWAYS or ALLOWED, down level JES2 members can NEVER again enter the MAS (even if LARGEDS is set to FAIL).

Getting to Large SPOOL volumes

1. **On a test system, test applications that access SPOOL by setting LARGEDS=ALWAYS and starting a SPOOL volume**
2. **Migrate to z/OS 1.7 JES2 on all MAS members**
3. **Wait for z/OS 1.7 JES2 to stabilize (no need to fall back)**
4. **$T SPOOLDEF to LARGEDS=ALLOWED**
5. **Stabilize with the new format of data areas**
6. **Start a large SPOOL volume**
   - Consider using SPOOL affinity to limit jobs using new SPOOL
7. **Once stabilized, drain old SPOOL volumes and migrate to new larger SPOOL data sets**
   - Clear SPOOL affinities if used for testing earlier

REMEMBER, once LARGEDS=ALLOWED is set, pre-z/OS 1.7 JES2s cannot be warm started even if set to LARGEDS=FAIL

This is the preferred migration path to large SPOOL data sets. It minimizes the risk to the system and provides a reasonable backout plan.

Exit/application Implications

- **Some applications may attempt to read/write SPOOL records**
- **Reading is supported using the SPOOL READ SSI**
  - Applications need to treat 4 byte SPOOL address as a token
  - Should not examine individual fields
- **Direct writing to SPOOL is not supported**
- **Applications that convert MTTR to BBCCHHR will need to updated to support new MTTtr SPOOL address format**
  - Example of conversion can be found in routine $EXCP in HASPNUC
- **Fields in the $DAS, $IOT, and $TGB (BLOB) are updated to 5 or 6 byte MQTs or MQTRs**
  - Examine data areas for any changes

If there are exits or applications that read or write the checkpoint directly, they may be impacted by the changes in this support. In particular, an application that converts the MTTR to BBCCHHR for use in CCWs needs to be updated. Code in HASPNUC can help you convert any code that you may have that does this.

Reader/NJE exit changes

- **TCP/IP NJE processing occurs outside the JES2 main task**
- **INTRDR processing also moving outside JES2 main task**
- **Main task exits no longer get control for TCP/NJE and INTRDR jobs**

- **New exits were added corresponding to current exits**
- **Additional main task exits have been defined**
  - When jobs are added to the job queue

- **Exits 36 and 37 will still be called but from a different address space**
- **Exit 8 will be called in some cases instead of exit 7**

- **Changes are INCOMPATIBLE with previous releases of JES2!!!!!**

The changes to NJE to implement TCP/IP will make it impossible to call the traditional HASPRDR exits in the JES2 main task. Similarly, changes to internal reader processing will also make it impossible to call the traditional HASPRDR exits in that environment. To address this, a new set of input processing exits has been defined. These exits will run in the user environment in the NETSERV address space. In addition, a new exit, exit 51, was defined in the main task when jobs change phase. Data can be passed to exit 51 from other exits. Exit 51 can be used as the ultimate end of input exit in the main task for all input sources.

In the case of exits 36 and 37, the exits will still be called, but they are called from a different address space.

For control block I/O, since the I/O is being done outside the main task, exit 8 instead of exit 7 will be called.

Reader/NJE exit changes *(Cont…)*

- **New exits will be added corresponding to these main task exits**
  - 2 – JCL job card
  - 3 – Job card accounting field
  - 4 – JCL and JES2 control (JECL) statement
  - 20 – End of input
  - 39 – NJE SYSOUT SAF rejection
  - 46 – NJE header transmit exit
  - 47 – NJE header receive exit
- **New exits will be defined for**
  - 51 – $QMOD, Job phase change
- **Other exits affected:**
  - 7/8– Control block I/O
  - 13 – TSO/E NETMAIL notify (to be deleted)
  - 36/37 – Pre and post SAF exit

This is a list of the exits affected. New exit numbers were defined for exits that need to be called outside the main task.

Reader/NJE exit changes *(Cont…)*

New Exit	Similar exit	Environ	Function
50	20	USER	End of input
51	*	JES2	$QMOD - job phase change
52	2	USER	Input processing - JOB card
53	3	USER	Input processing - Accounting field
54	4	USER	Input processing- JCL/JECL
55	39	USER	NJE SAF rejection
56	46	USER	NJE header/trailer transmit
57	47	USER	NJE header/trailer receive

- All exits (new and changed) will be passed XPLs
- XPLs for new and similar exit will be the same
- New data areas will contain former PCE/DCT fields
  - Passed to both exits
- Old exits will be passed same data as in previous releases

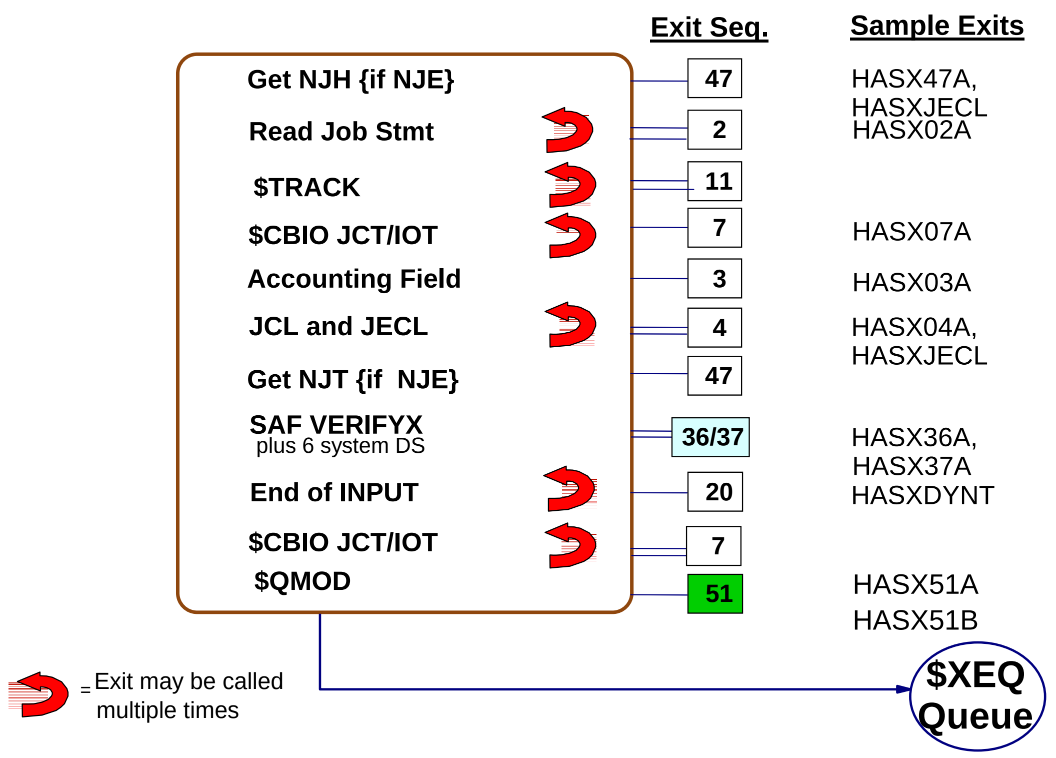
This is a list of the new exit numbers, the similar old exit, and the environment of the new exit. All exits will be passed XPLs. Existing exits will have XPLs available as well as the current input registers. The XPLs for the new and old exits will have the same data (but separate mappings). Some data areas that were in PCEs will be moved to new data areas that will be common to both environments. The XPL will formalize some of the interfaces and simplify some of the tasks commonly performed in each exit (based on customer and vendor feedback).

Reader/NJE exit changes *(Cont…)*

- **Old style exits (2, 3, 4, 20, 39, 46, 47) still used for:**
  - Local card readers
  - RJE readers
  - SNA and BSC NJE transmitters and receivers
  - Spool Offload transmitters and receivers
- **New style exits (52, 53, 54, 50, 55, 56, 57) used for:**
  - Batch Internal readers
  - STC and TSU internal readers
  - TCP/IP NJE transmitters and receivers
- **New exit 51 receives control for all phase changes:**
  - Job moves from $INPUT to $XEQ, etc.
  - Job requeued for execution

The old exits are still used for all but internal readers and NJE/TCP. The new exits are used for NJE/TCO and internal readers. Exit 51 is a main task exit that gets control as jobs move from one phase to the next. For NJE/TCP and internal reader, this is the first main task exit for the job.

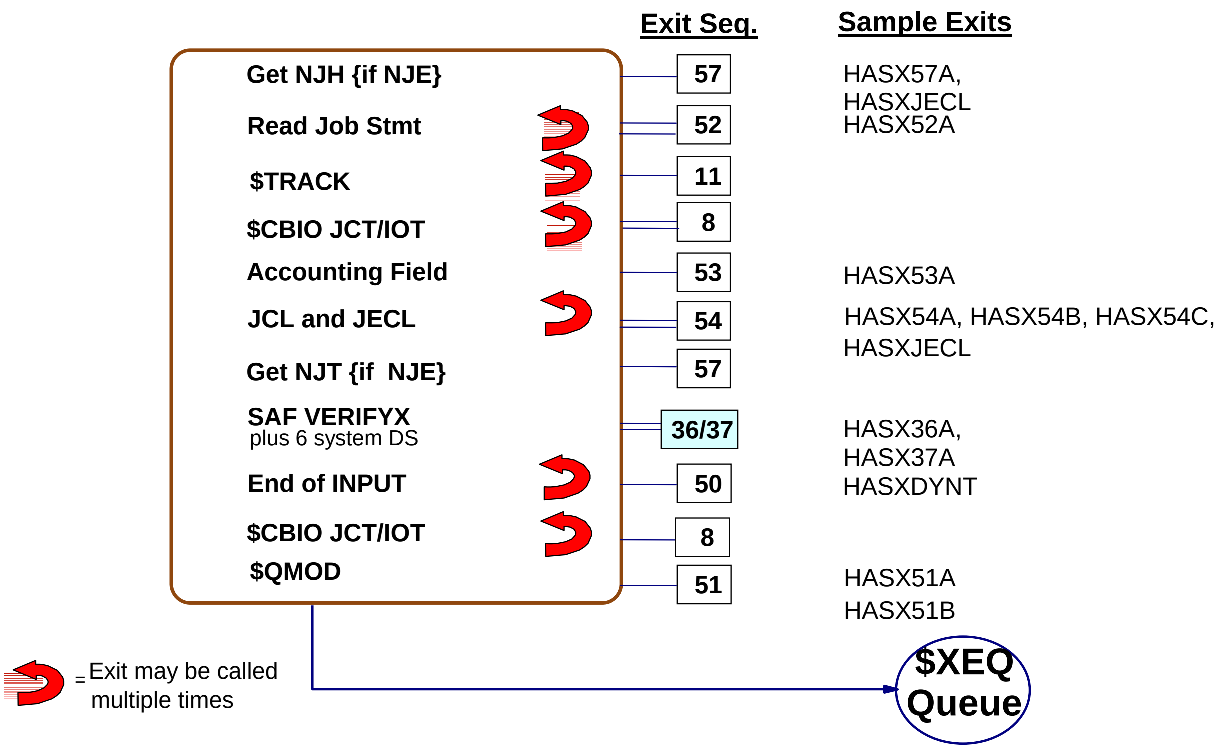
JES2 Input Processing – Main Task



This is the sequence of exits called for jobs coming into the system under a device driven by the main task – RJE or local reader, SNA or BSC NJE device, or spool reload. This is the same sequence of exits that has always received control for input devices. The interfaces to exits 2, 3, 4, and 20 have changed somewhat.

Exit 51 is new and is driven from the main task when the job is moved from the $INPUT queue to the conversion queue.

JES2 Input Processing – USER environment



These are the exits for devices driven in the user environment. The exit numbers are different, but exits are driven at the exact same points in processing as their main task counterparts. Except for the environmental considerations (R11=HCT vs HCT, R13=PCE vs. save area), the data passed to these exits is also the same as their main task counterparts.

Note that exit 51 is still the last exit driven for the job.

Exit 13 (NJE Notify)

- **$EXIT 13 has been deleted**
  - $HASP548/$HASP549 no longer issued from SYSOUT receiver
    - Too early anyway
    - No way to force for locally created output
  - $HASP548/$HASP549 now issued during OUTPUT processing
    - When job is actually available to TSO RECEIVE
    - Can now force message out for locally created output
- **$EXIT 13 functionality replaced by:**
  - **NJEDEF MAILMSG=YES/NO** (since SP 4.3.0!!!)
  - $EXIT 47 (SYSOUT receiver DSH reception)
    - Process data set header fields in SYSOUT receiver
  - $EXIT 40 (JOE creation)
    - Flags in $XPL to influence whether messages issued
    - Flag in PDDB to influence job eligibility for message

In this release, JES2 is addressing a long standing complaint about the message issued when SYSOUT for a TSO user is received. Prior to this release, the message was issued early in processing the SYSOUT data set that was being received. If the SYSOUT data set was large and the TSO user did a receive after seeing the message, it is possible that the data set may not yet be available for processing.

The notify processing was moved from SYSOUT reception processing to OUTPUT/SPIN processing. This ensures that the message is not received before the output is ready for processing. However, exit 13 no longer made sense in this environment. As a result, and because it is unlucky, we deleted exit 13. The function of the exit can be replaced by the existing external NJEDEF MAILMSG and new function added in exit 40.

Exits 2 and 52 (Job card scan)

- **Exit 2 (JES2 main task)**
- **Exit 52 (USER environment)**
- **Exit now passed XPL in register 0**
  - Pointers to JQE, JCT, JRW, card image
  - Pointer to composite job card in buffer with ALL specified operands
    - Don’t have to deal with continuations or parameters spanning cards
  - Indicators for last operand is on card, unfinished quote at end of card, card is a continued comment and last card in statement
- **Output fields in XPL**
  - RJCB chains – cards to add before or after current statement or after current card
  - Override job class for job
  - Flags to cancel or purge current job

New exit 52 gets control at the same point in processing as the existing exit 2. The exit has been enhanced to simplify parsing the current JCL statement and the adding new JCL to the job stream. Additional information is passed to help exits determine if it is safe to add a card after the current card or not (are we in the middle of a quoted string for example).

Exits 4 and 54 (JCL/JECL card scan)

- **Exit 4 (JES2 main task)**
- **Exit 54 (USER environment)**
- **Exit now passed XPL in register 0**
  - Pointers to JQE, JCT, JRW, card image
  - Pointer to composite card in buffer with ALL specified operands
    - Don't have to deal with continuations or parameters spanning cards
  - Indicators for last operand is on card, unfinished quote at end of card, card is a continued comment and last card in statement
  - Label and verb, JCL/JECL card indicator
- **Output fields in XPL**
  - RJCB chains – cards to add before or after current statement or after current card
  - Flags to cancel or purge current job

Essentially, the XPL for exits 4 and 54 are the same as the XPL for exits 2 and 52. This was done to simplify the logic in JES2 as well as the processing in the exit. It is possible to code one service routine that handles all JCL (JOB card as well as JCL/JECL cards) using just the exit 2 version of the $XPL. The indicator byte (X0xxIND) can be used to determine if this is a job card or not.

Exit 2, 4, 52, 54 Statement Buffers

- **JES2 will parse all operands off a statement (and it's continuations) and pass this to the exits. For example:**

```
//TEST     JOB (12345),'SYSTEM PROGRAMMER',
//             MSGLEVEL=(1,1),CLASS=A  Comment 1   X
//             This is a comment
```

- **The statement buffer passed to the exit will contain**

```
(12345),'SYSTEM PROGRAMMER',MSGLEVEL=(1,1),CLASS=A
```

- **This can then be parsed using**
  - $SCAN if JCL rules not to be used
    - Some customers currently use $SCAN to parse their own JECL
  - RCARDSCN/$STMTTAB if you want JCL rules
    - Preferred method for JCL statements
  - Each has its own advantages
    - Examples of both can be found in **HASX54A** and **HASX54B**

Before calling exits 2, 4, 52, or 54, JES2 will read the entire JCL statement and parse all the operands off the statement. These operands are assembled into a single buffer called the statement buffer. This buffer greatly simplifies parsing of keywords from the JCL cards since the exit does not need to implement the JCL rules for continued statements.

JES2 also provides a service, RCARDSCN to parse the individual operands and process each one. The service only handles the parsing, it does not do any conversion of values or setting of fields. The input to RCARDSCN is a table of $STMTTABs that control the parsing and specifies routines to be called.

Parsing a statement - example

- **This is an example of how to parse the SCHENV= operand off the JOB card (from HASX52A):**

```
RSCHKEYL $STMTTAB      TABLE=USER
         $STMTTAB      KEYWORD='SCHENV',ROUTINE=PROCSCHE
         $STMTTAB      KEYWORD=UNKNOWN   Ignore unknown keyword
         $STMTTAB      TABLE=END
```

- **The call to RCARDSCN is then:**

```
         LARL    R0,RSCHKEYL    Get JOB card key list
         $CALL   RCARDSCN,             Call card scan service
                 PARM0=(R0),           JOB card keylist
                 PARM1=JRW,            JRW address
                 OKRET=RJCLRET         Error, pass on to caller
```

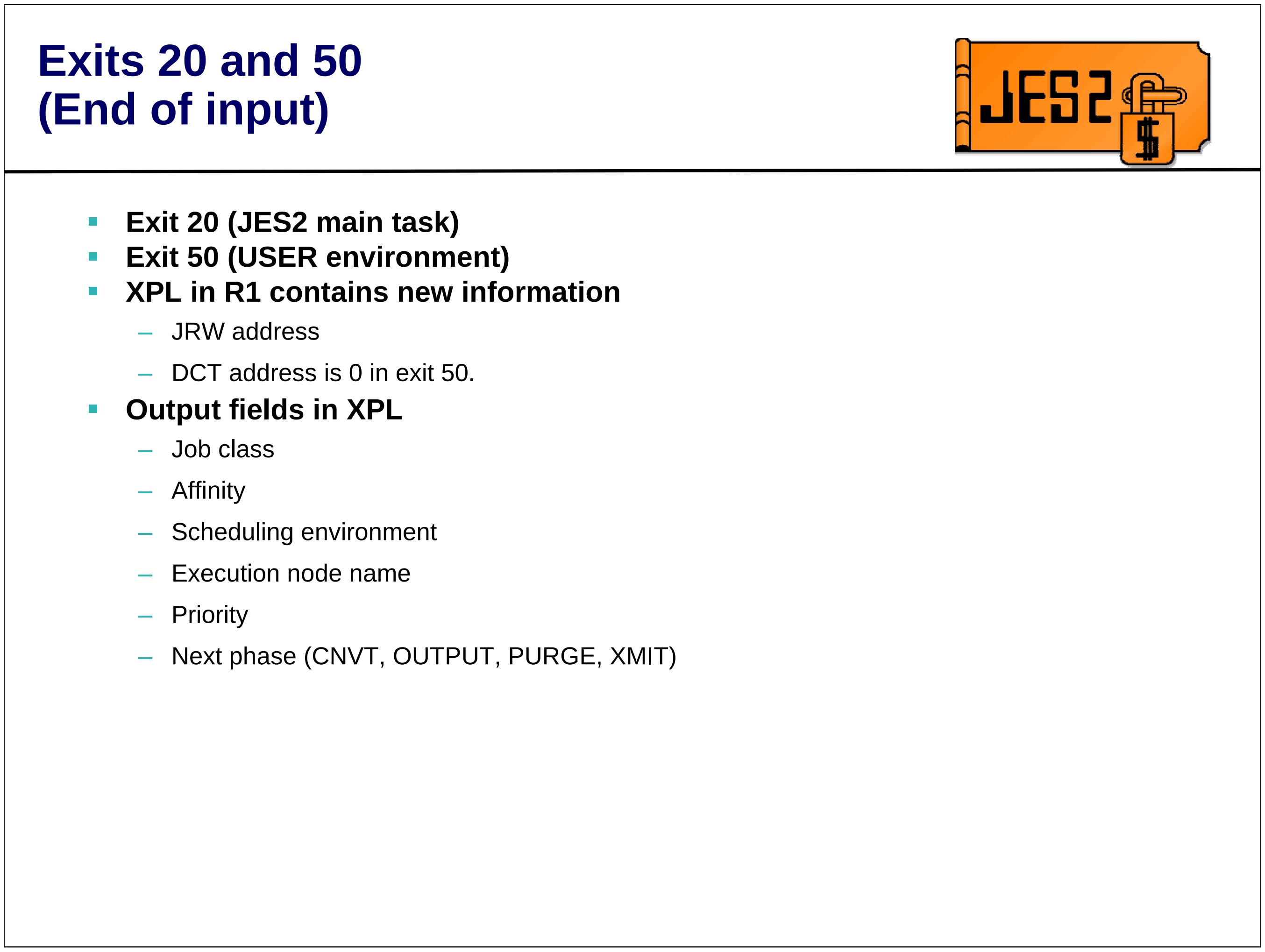
- **When the SCHENV keyword is encountered, routine PROCSCHE is called.**
  - Routine processes and validates keyword and returns
  - Can return a specific error message for specific failures
- **All keywords not in table (i.e. unknown) are ignored**

Here’s an example of how a customer could parse parameters off the job card using RCARDSCN. The JRW is already set up with everything necessary to parse the statement, so the only input parameters are the JRW and the table to use for parsing.

Exits 3 and 53 (Job card accounting scan)

- **Exit 3 (JES2 main task)**
- **Exit 53 (USER environment)**
- **Exit now passed XPL in register 0**
  - Pointers to JQE, JCT, JRW, accounting string
  - R0 was accounting string length prior to z/OS 1.7
    - Load from X003ACTL instead
    - Load address of string from X003ACCT
- **Output fields in XPL**
  - Flags to bypass default accounting field scan
  - Flags to cancel or purge current job

Exits 3 and 53 get control at the same point in processing as exit 3 did in previous releases. The exit is now passed an XPL with all the information needed to process the accounting string.



Exits 20 and 50 (End of input)

- **Exit 20 (JES2 main task)**
- **Exit 50 (USER environment)**
- **XPL in R1 contains new information**
  - JRW address
  - DCT address is 0 in exit 50.
- **Output fields in XPL**
  - Job class
  - Affinity
  - Scheduling environment
  - Execution node name
  - Priority
  - Next phase (CNVT, OUTPUT, PURGE, XMIT)

Exits 20 and 50 get control at the same point in processing that exit 20 got control in previous releases. The XPL that was passed to the exit was expanded to include new override fields that simplify setting attributes of the job (you do not need to know where the fields are stored, just test or set the fields in the XPL). Though in the past, exit 20 was the last input processing exit, now exit 51 will give exit writers one last chance in the JES2 address space to alter properties of the job.

Exits 46, 47, 56, 57 (NJE header/trailer xmit/receive)

- **Exits 46, 47 from main task (SNA, BSC, spool offload)**
- **Exits 56, 57 from USER environment (TCP/IP)**
- **Additional input to exit**
  - NJEWORK address (JRW, JTW, SRW, or STW, depending on exit and device type)
  - DCT address is 0 in 56 and 57

Exits 46, 47, 56, and 57 get control at the same point in processing as exits 46 and 47 did in previous releases. The XPL was updated to pass the pointer to the appropriate local work area (JRW, JTW, SRW, or STW).

Exit 49 ($QGOT)

- **$EXIT 49 enhancements**
  - Now called for $SJ command processing and $SJ selection
    - X049IND reflects call type
    - Should not commit resources to job in $SJ command processing call (X049IND=X049SJOB)
    - $QGET parameter address (X049QGT) is zero in this case
  - Response flag to bypass duplicate job checking
    - For exits that need to do it themselves

Exit 49 was updated in this release to get control for $SJ (start job) processing. $SJ processing has 2 phases. The first occurs under the command processor and it validates that the job is a candidate for start job processing. If so, a call is made to WLM to start an initiator and request the job being started. At some later point (and perhaps on another member of the MAS) the WLM initiator calls JES2 to request the job. At this point, in the 2nd phase, checks are made a second time to ensure the job is still eligible for $SJ processing and if so, the job is passed to the initiator. Exit 49 is now called in both phases of $SJ processing. Failing a job in the first phase results in a message back to the operator console where the command was issued that the job command cannot complete. Failing a job in phase 2 only results in a message to the console were the initiator was started, that the job was not selected.

The important incompatibility is that the $QGET parameter list in the XPL is zero in this new case. This may cause errors for existing exit 49s that expect this value to be non-zero.

An additional response bit was added (X049NDUP) to indicate to JES2 to bypass duplicate job name checking. It is assumed that the exit has performed the needed checks. This is useful if the exit has different rules for duplicate job name holds than the standard JES2 rules.

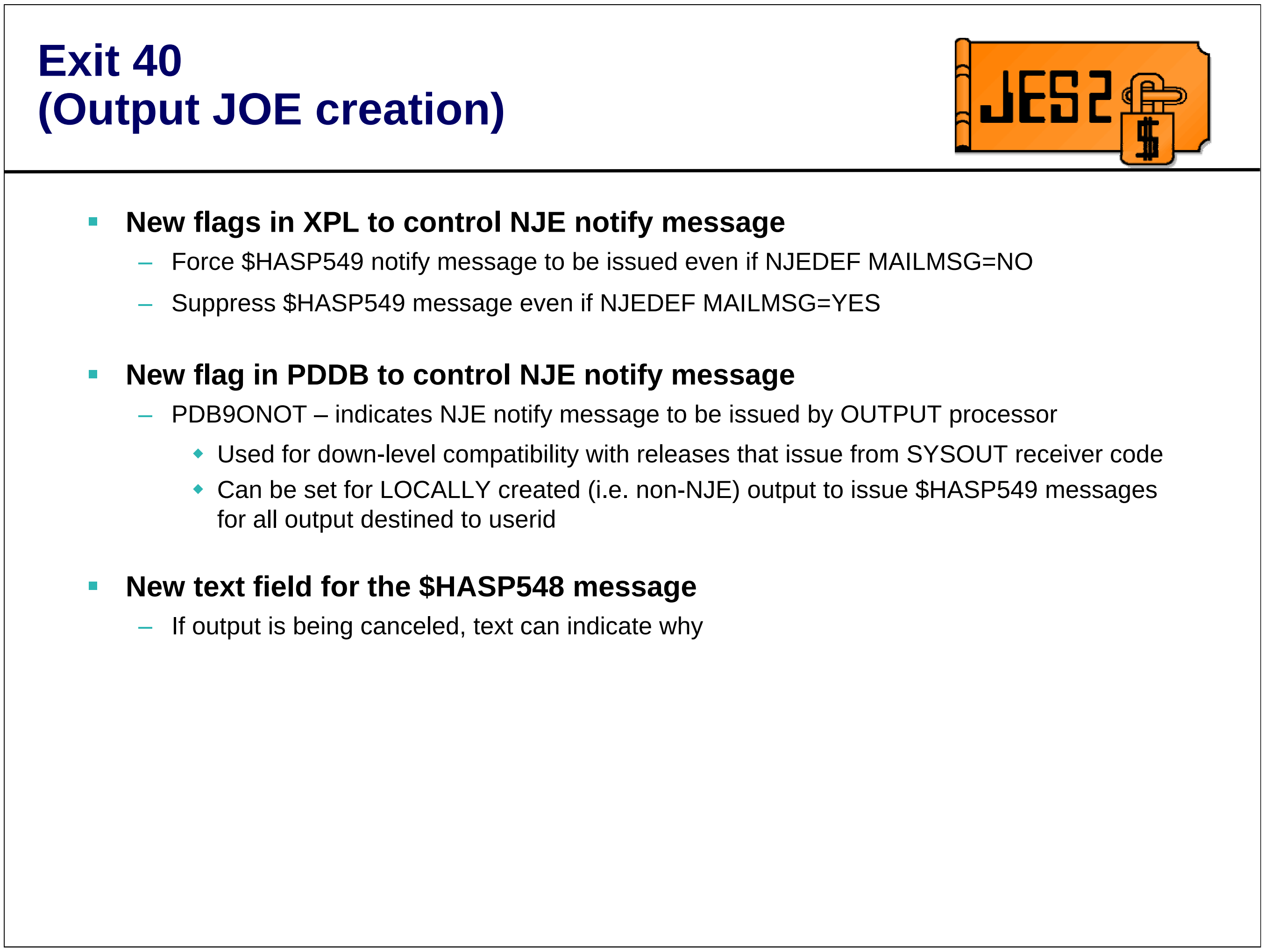
Exit 51 ($QMOD)

- **New exit**
- **Receives control when:**
  - Job moves from one phase to another ($INPUT to conversion)
  - Job is requeued for execution
- **XPL in register 1**
- **Input fields**
  - JQA address
  - JCT address (if moving from $INPUT or $RECEIVE)
    - JCT will NOT be written out after exit
  - New and old queue types
- **Output fields**
  - New queue type
  - JOBCLASS, SCHENV, or SYSAFF overrides
  - Re-queue job for execution (or prevent re-queue)

Exit 51 is a new exit that gets control whenever a job changes phase or for the special case of a job exiting execution but being re-queued for execution. The exit does not get control when an operator changes a job class, priority, service class, etc. It only gets control when a job change phases.

The exit can be used for tracking jobs as they move from one phase to another or to alter the job's characteristics or next phase of processing. When altering a jobs next phase, the exit cannot make a job go to an earlier phase in processing (for example, a job exiting the OUTPUT phase cannot be queued to execution).

As with other XPLs, a number of overrides are provided in exit 51. The exit is passed the current values of these fields and can alter them by setting the new value in the XPL.



Exit 40 (Output JOE creation)

- **New flags in XPL to control NJE notify message**
  - Force $HASP549 notify message to be issued even if NJEDEF MAILMSG=NO
  - Suppress $HASP549 message even if NJEDEF MAILMSG=YES
- **New flag in PDDB to control NJE notify message**
  - PDB9ONOT – indicates NJE notify message to be issued by OUTPUT processor
    - Used for down-level compatibility with releases that issue from SYSOUT receiver code
    - Can be set for LOCALLY created (i.e. non-NJE) output to issue $HASP549 messages for all output destined to userid
- **New text field for the $HASP548 message**
  - If output is being canceled, text can indicate why

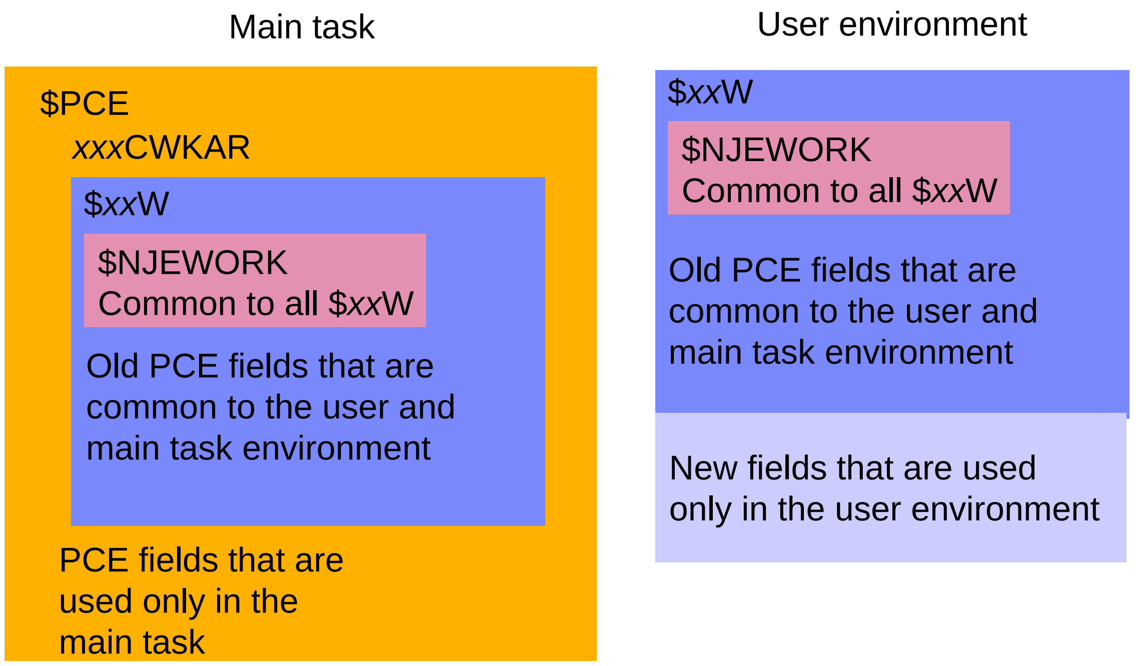
Exit 40 was enhanced to allow installation to determine if a mail notification message should be issued. It can cause a message to be issued when it was not going to be or suppress a message that would have been issued. A new indicator in the PDDB exists to inform the exit whether the notify message was issued or not.

This exit always had the ability to delete the SYSOUT being received. It now can also force the HASP548 message to be issued in this case and specify a reason text (20 bytes) that indicates why the SYSOUT is being canceled.

NJE Work areas (JRW, JTW, SRW, STW)

- **NJE Work areas**
  - Common mapping mapped by $NJEWORK macro
    - Prefix to generate common mappings
      - $JRW invokes $NJEWORK PREFIX=JRW, etc.
    - IPCS definitions included in $NJEWORK macro
  - Work areas reside in:
    - PCE work areas of subdevice PCEs for BSC, SNA, TCP/IP
    - User address space for internal readers (JRW)
    - NETSRV address space for TCP/IP subdevices
      - Both PCE area and NETSRV address space area exist for NJE/TCP transmitters and receivers
      - Setting a field in one does not imply it's set for the other
  - Contains:
    - Common anchors used for JCT, JQE, JOE, PDDB, headers, etc.
    - Work fields (NJEDBLE etc.)

PCE/User environment data structures


Main task
User environment
$PCE
*xxx*CWKAR
$*xx*W
$NJEWORK
Common to all $*xx*W
Old PCE fields that are common to the user and main task environment
PCE fields that are used only in the main task
$*xx*W
$NJEWORK
Common to all $*xx*W
Old PCE fields that are common to the user and main task environment
New fields that are used only in the user environment

The PCE work areas have been broken up into fields that are common to all NJE/TCP devices, ones common to a particular type of device, and ones that at needed only in the main task. As a result, what used to be one data area is now 2 DSECTs and 3 macros. For example, fields that were in the $RDRWORK PCE work area are now in either $RDRWORK (fields only needed by the main task), $JRW (fields needed for main task and user environments) or $NJEWORK (fields common to all NJE devices and internal readers). The prefix for fields in the $RDRWORK PCE has remained RDW. But fields in the $JRW and $NJEWORK mapped in the $JRW have a prefix of JRW. Most of the time, you can translate RDWxxxxx fields directly to JRWxxxxx fields. To access the $JRW in the main task, the following USING should work (assuming addressability to the $PCE), USING JRW,RDWCWKAR

Another example is the $NSRWORK data area. Fields that were in it are now in $NSRWORK, $SRW, and $NJEWORK. The prefix for fields in these work areas remained SRW.

Common mapping of areas
- example

- **$RDRWORK**
  - Defines **RDWCKAR DC XL(JRWCLEN)**
    - (length of commonly defined JRW)
    - Area mapped by $JRW
  - Additional fields in PCE work area used only by main task
- **$JRW generates common area using NJEWORK macro**
  - **$NJEWORK PREFIX=JRW,SYSOUT=NO**
  - Additional fields defined after NJEWORK mapping
    - Common to both environments but only apply to job receivers
    - JRWCLEN includes these fields
  - Additional fields defined after JRWCLEN
    - Used only in user environment
    - JRWLEN is total length
- **Similar techniques used by other transmitters and receivers**

Other changes for user environment exits

- **HCT, PCE, and DCT are not available**
  - Not in JES2 address space
  - Not serialized for updates
  - JES2 address space may be down!
  - For most uses, alternatives are available
    - Corresponding HCCT fields may exist
      - Use CCTBLNKS, CCTZEROS instead of $BLANKS, $ZEROES, for example
    - Corresponding fields in JRW, SRW, JTW, STW may exist
      - Use JRWDEVN, JRWDEVID, etc. instead of DCTDEVN and DCTDEVID
      - Most interesting PCEWORK fields now exist in JRW, SRW, JTW, STW
        - RDWxxxxx fields are now JRWxxxxx
    - Work fields in JRW, SRW, JTW, STW
      - JRWDBLE, JRWDBLE1, etc.
    - JQA and JCT both available
      - $DOGJQE unnecessary in user environment

Techniques for writing common exits

1. **Replicate code in two places (e.g. exit 2 and exit 52)**
   - Must maintain 2 sets of code
2. **Use COPY or macro to include common code from a single source**
   - Conditional assembler logic to isolate environment-specific code
3. **Invoke common routine from both exits**
   - Use “glue” routine to get from main task routine to common routine
   - Can be placed in same load module as common routine
   - Locate common routine using LOCENTRY service
   - Use XPLID or TCB/ASCB addresses and $ENVIRON to isolate environment-specific code
4. **Use $EXIT to invoke common routines**
   - All routines must support multiple environments

Here are 4 techniques to simplify the writing of “dual” exits. Many of these techniques can be seen in sample exits that are provided in z/OS 1.7.

Example 1 (COPY code)

- **Create COPY code to invoke from either environment**
- **Use conditional assembly logic to isolate environment-specific code**

```
       AIF ('&ANVIRON' EQ 'USER').USER
       <generate main-task specific code here>
       AGO  .COMMON
.USER  ANOP
       <generate user-environment specific code here>
.COMMON  ANOP
```

- **Can also use assembler variables to deal with environment-sensitive field**
  - e.g. set &BLANKS to $BLANKS in JES2 environment, CCTBLNKS in user environment

Using COPY code provides one set of code that implements the function. This is good for code that has little dependencies on the environment in which it runs, but cannot be easily made common.

Example 2 (common routine)

- **Create common routine using environment USER,ANY**
- **Create main task stub routine that sets R11=HCCT (via $ENVIRON) and invokes common routine**
  - Find address of common routine by either:
    - Placing stub routines in same load module as common routine
    - Using LOCENTRY service to locate the entry point (from MITs/MITEs of LOADed modules)
    - Invoking $EXIT directly
- **Isolate environment-specific code by either:**
  - Checking XPLID for USER vs. JES2 exit number
  - Checking R13 save area eyecatcher for “PCE “
  - Comparing ASCB and TCB address for JES2 main task to PSAAOLD/PSATOLD

This technique can be a bit more work, but the finished code is often easier to understand.

(USER,ANY) assembly environment

- **New assembly environment**
  - $MODULE **ENVIRON=(USER,ANY)**
  - $ENVIRON SET,**ENVIRON=(USER,ANY)**
  - Influences $SAVE/$RETURN and $STORE/$RESTORE services
    - R11 = HCCT on entry
    - If R13 = PCE address, uses main task $SAVE/$RETURN services
    - If not, creates PSV chained to current save area
    - No linkage stack entry created (allows $WAIT if JES2 main task)
  - Allows for common coding of routines which must run in both main task and user environments
  - Used extensively by:
    - JES2 services such as $SCAN, $BLDMSG, NJE service routines, Input processing
    - Sample exits
- **Session 2665, JES2 Exits overview, Thursday 1:30**

As part of implementing NJE/TCP, many of the existing service routines that were used for BSC and SNA needed to be made available for use in the new NETSERV address space. This included the services for input processing. To simplify the processing, a new JES2 assembly environment was created called USER ANY. In this environment, the $SAVE and $RETURN macros use either the main task or user save services based on the environment that is currently active. This allows access to the PCE in the main task as well as access to services such as $WAIT. In this way, general purpose code can be written to run in either environment by using a few special purpose, environment sensitive routines (such as FREEJCT) that perform environment specific tasks. This minimizes the coding and maintenance effort. This is a very effective way to implement a single exit routine that can run in multiple environments. A number of sample exits use this technique.

Save areas

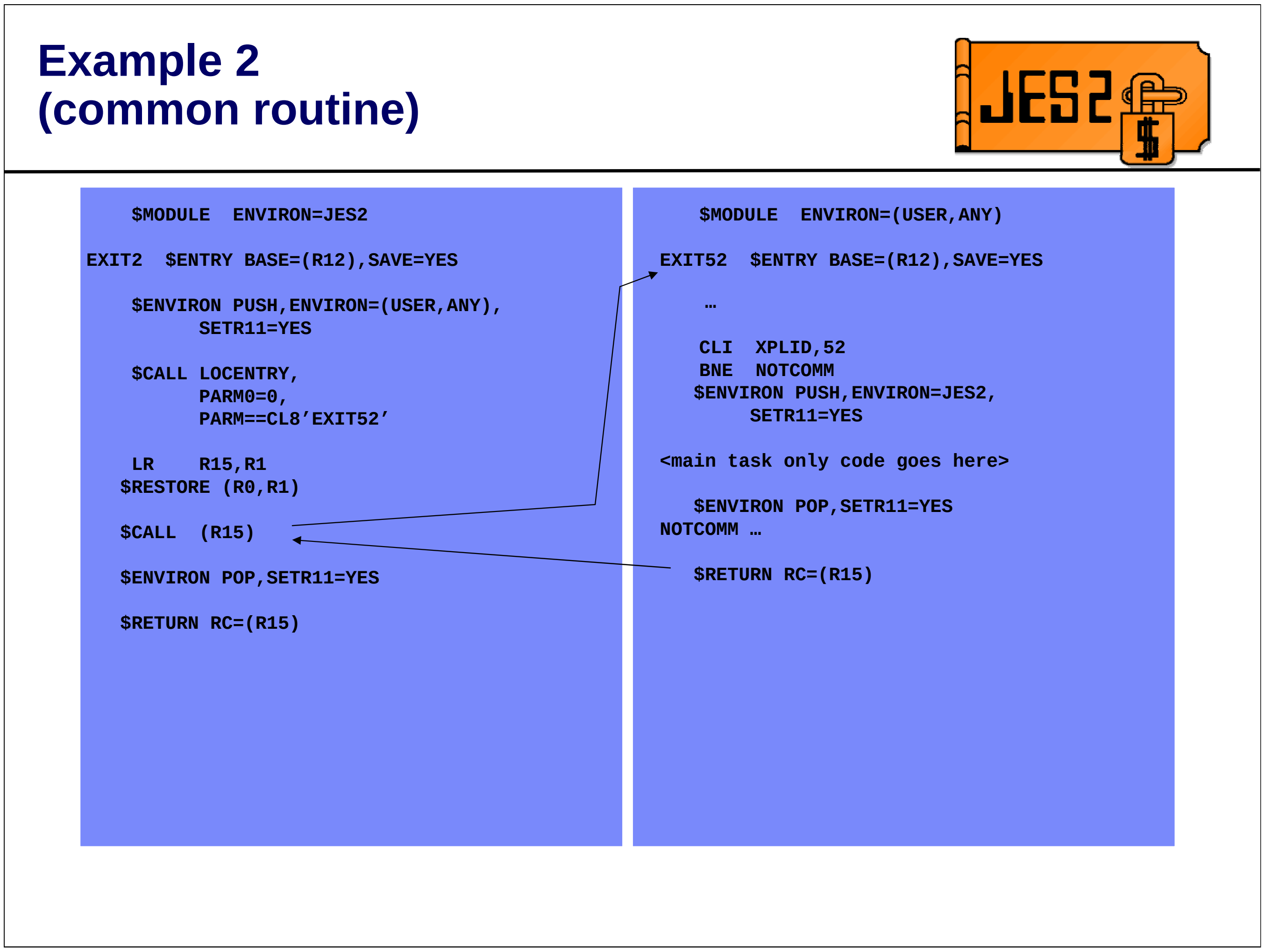
- **ENVIRON=(USER,ANY) saves generate PSVs, as do main task saves**
  - $STORE/$RESTORE can be used
- **Address of save area chain in PSV in front of TRE**
  - Same offset as in main task for PSV in front of PCE
- **ENVIRON=USER generates linkage stack and TRX entries**

- **ENVIRON=(USER,ANY) and ENVIRON=USER saves can be mixed in any order**
  - Not a problem at run-time as long as routines play by the rules
    - Routines using (USER,ANY) save must use (USER,ANY) return
    - Routines using USER save must use USER return
  - Sorting them out in a dump is another story
    - Use linkage stack pointers in TRX and PSV

$STORE/$RESTORE changes

- **$SAVE/$RETURN always save ARs and full 64-bit general purpose registers**
- **$STORE/$RESTORE in JES2 or (USER,ANY) environment:**
  - $STORE/$RESTORE (register) – operates on full 64-bit reg
  - $STORE/$RESTORE REGS=(register) – full 64-bit register
  - $STORE/$RESTORE AREGS=(register) – access register
  - $STORE/$RESTORE QREGS=(register) – full 64 bit register and access register
- **Example of macro expansion**

```
         $STORE QREGS=(R4)          SAVE NEW(?) BUFFER ADDRESS
+         L     R13,PCELPSV-PCE(,R13)  Get addr of save area
+         ST    R4,PSVR4-PSV(,R13)        Save register
+         STMH  R4,R4,PSVHR4-PSV(R13)      and high half
+         L     R13,PSVARPTR-PSV(,R13)  Point to AR area
+         STAM  AR4,AR4,PSVAR4-PSVAREGS(R13)  Store AR
+         L     R13,PSVARCHN-PSVAREGS(,R13)  Back to save area

+         L     R13,PSVPCE-PSV(,R13)  Restore PCE address
```



Example 2 (common routine)

```
    $MODULE  ENVIRON=JES2

EXIT2  $ENTRY BASE=(R12),SAVE=YES

    $ENVIRON PUSH,ENVIRON=(USER,ANY),
          SETR11=YES

    $CALL LOCENTRY,
          PARM0=0,
          PARM==CL8'EXIT52'

    LR    R15,R1
   $RESTORE (R0,R1)

   $CALL  (R15)

   $ENVIRON POP,SETR11=YES

   $RETURN RC=(R15)
```

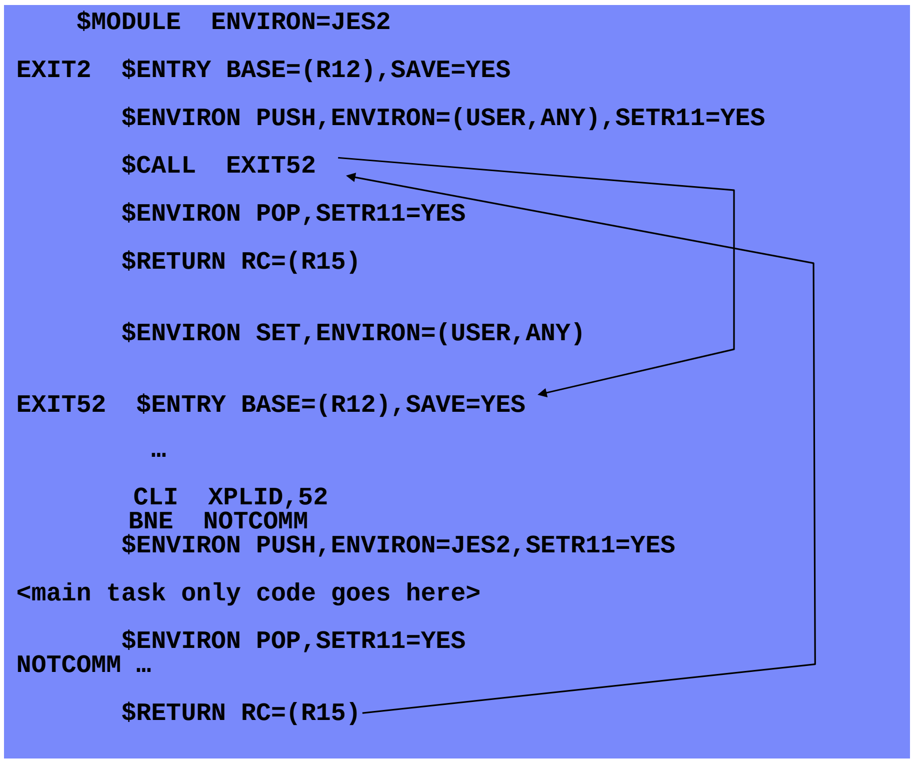
```
   $MODULE  ENVIRON=(USER,ANY)

EXIT52  $ENTRY BASE=(R12),SAVE=YES

    …

   CLI  XPLID,52
   BNE  NOTCOMM
  $ENVIRON PUSH,ENVIRON=JES2,
       SETR11=YES

<main task only code goes here>

  $ENVIRON POP,SETR11=YES
NOTCOMM …

  $RETURN RC=(R15)
```

The USER,ANY environment allows you to use main task and user environment service as needed based on the current runtime environment.

The LOCENTRY service is used by some of the sample exits to allow the exit to be shipped and loaded separately. The LOCENTRY service is an expensive service to call each time an exit is invoked. At the very least, it should be called during JES2 initialization ($EXIT 24) and the address of the routine stored in a UCT field.

A better way is to package both entry points in the same load module in CSA.

Example 3
(common routine)



```
    $MODULE  ENVIRON=JES2

EXIT2  $ENTRY BASE=(R12),SAVE=YES

       $ENVIRON PUSH,ENVIRON=(USER,ANY),SETR11=YES

       $CALL  EXIT52

       $ENVIRON POP,SETR11=YES

       $RETURN RC=(R15)


       $ENVIRON SET,ENVIRON=(USER,ANY)


EXIT52  $ENTRY BASE=(R12),SAVE=YES

         …

        CLI  XPLID,52
        BNE  NOTCOMM
       $ENVIRON PUSH,ENVIRON=JES2,SETR11=YES

<main task only code goes here>

       $ENVIRON POP,SETR11=YES
NOTCOMM …

       $RETURN RC=(R15)
```

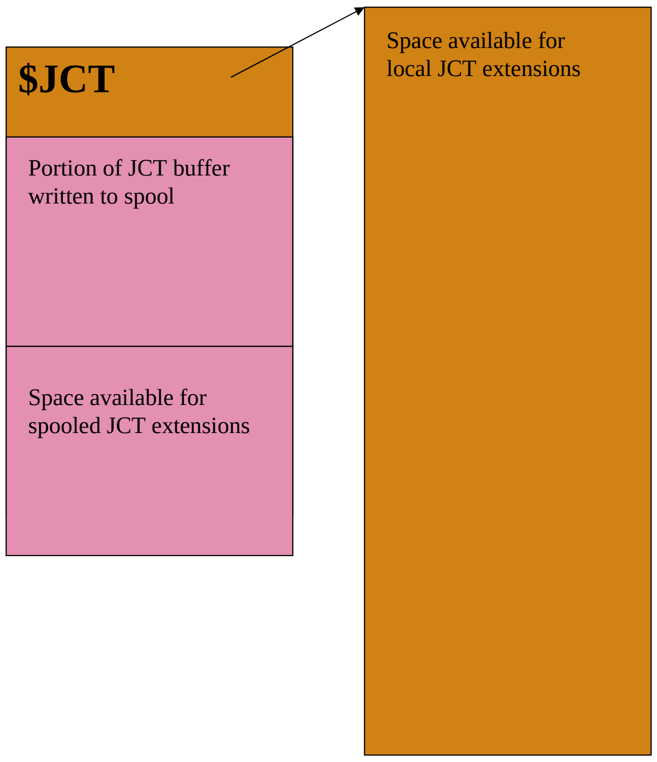
This example shows the preferred technique, packaging both the stub routine and the common routine in the same load module. Sample exits HASXJECL and HASXDYNT both use this technique.

Local JCT extensions

- **Currently, the $JCTXADD service can be used to create SPOOLed JCT extensions**
- **New support adds a LOCAL, non-SPOOLed extension**
  - Available in limited environments
    - Input processing (RDR, NJE job receivers and INTRDRs)
    - NJE SYSOUT receivers
  - Does not use limited JCT space for data that does not need SPOOLing
  - Up to 8K of data can be associated with the JCT
  - Data is lost when JCT is freed
  - Can be used to pass data from one exit to another
    - Pass data between exits 2/52, 3/53, 4/54, 20/50, 47/57 and 51
    - Transported to JES2 address space from NETSERV/INTRDR

Local JCT extensions can be used to associate data that does not need to be SPOOLed with a JCT. This data will persist until the JCT is freed. You can have up to 8K of data in local extension in addition to the SPOOLed extensions. Only limited environments support local extensions at this time. These include the traditional HASPRDR (SNA/BSC NJE, RJE, SPOOL reload, card readers), traditional SYSOUT receivers, NETSRVs (jobs and SYSOUT receivers) and internal readers. These local extensions were put in place to assist in communicating information from the user environment exits to the $QMOD phase change exit 51. But they can be used to communicate between any of the exits where the JCT is passed.

Local JCT extensions


$JCT
Portion of JCT buffer written to spool
Space available for spooled JCT extensions
Space available for local JCT extensions

- **Spooled $JCT extensions**
  - End of JCT buffer
  - Always written to spool
  - Passed between phases of job
- **Local $JCT extensions**
  - Separate 8K private storage buffer
  - Currently supported only for jobs in “input” phases
  - Do not survive to next phase
  - Passed from USER address space to JES2 address space via $JQESERV

Using Local JCT extensions

- **Specify LOC=LOCAL on $JCTXADD macro**
  - Adds to non-spooled area instead
  - LOC=SPOOL is the default
  - Same extension TYPE= and MOD= not allowed in both spooled and non-spooled areas

```
$JCTXADD JCT=JCT,           Add a JCT extension                C
         TYPE='SAMP',        Type                               C
         MOD=(R3),           Modifier determined above          C
         LOC=LOCAL,          In local storage only              C
         LENGTH=(R2),        Length determined above            C
         FOUND=PJXTERRX,                                        C
         ERRET=PJXTERRR
```

- **Use $JCTXGET, $JCTXEXP, and $JCTXREM as with spooled extensions**
  - Services search both areas and process appropriate extension

$SCAN from non-main task environments

- **JES2 $SCAN services are now available outside the main task**
  - USER, SUBTASK and FSS environment supported
- **Includes $BLDMSG service**
- **Can be used from new input service exits to parse JECL cards (for example)**
- **User and dynamic tables supported**
  - Tables must be in CSA if called from user environment
  - CCTMGTP table pair in HCCT for dynamic BLDMSG tables outside JES2 address space
- **CB=HCT, CB=PCE not available in USER environment**
  - May require changes to $SCANTABs and $SCAN calls
  - Can pass a control block address to SCAN via CBADDR= parameter
  - For example, to reference JQE fields in exit 54:
    - Specify CBADDR=<jqe address> on $SCAN call, CB=PARENT on $SCANTAB

    or
    - Specify CBADDR=<jrw address> on $SCAN call, CB=PARENT and CBIND=(JRWJQE,JRW,L) on $SCANTAB

    or
    - Specify CBADDR=<xpl address> on $SCAN call, CB=PARENT and CBIND=(X054JQE,XPL,L) on $SCANTAB

The JES2 $SCAN services have been updated to support being called from outside the JES2 main task which includes being called from user environment exits and the use of the $BLDMSG services. This allows the $SCAN services to be used to parse basic JECL statements from the new input services exits in the USER environment.

The updated services support dynamic tables pair processing. Separate tables exist for user environment user of the $SCAN service (as well as the $BLDMSG service).

Samples

- **Samples created for all new exits**
- **Old exits 2, 3, 4, 20, 39, 46, 47 invoke new common code in exits 52, 53, 54, 50, 55, 56, 57**
- **Old exits functionally migrated to new exit points**
  - Changed to use HCCT fields instead of HCT fields (e.g. CCTBLNKS instead of $BLANKS)
  - Made re-entrant (HASX03A was NOT!)
  - Environment-sensitive code (HASX55A)
- **New exits to provide “useful” function**

JES2 z/OS 1.7 has shipped samples for all new exits. These samples can be used as examples or in some cases used as is to implement functions that are commonly done in exits. However, remember samples are not type 1 code and to not have the level of support that would normally find for JES2 code.

Sample exits in z7

- **“Useful” functions provided**
  - **/*AFTER, /*BEFORE, /*CNTL JECL statements**
    - Uses local JCT extensions and JQA extension to store values from JECL cards
    - Two different methods of parsing statements
      - RCARDSCN (HASX54A)
      - $SCAN (HASX54B)
    - Code in Exit 49 and 51 to manage job dependencies and resource counts
    - Code in exit 20/50 to copy data from local JCT extension to JQA
    - Intended as starting point for migration of existing mods
    - Code in HASX04A, HASX20A, HASX50A, HASX54A, HASX54B, HASX49B, HASX51B

Sample exits in z7

- Force job to convert on system where SCHENV available**
  - Exit 2/52 to parse SCHENV from job card
  - Exit 20/50 to force scheduling environment
  - Exit 49 to check SCHENV availability for converter $QGET
- **Force job to run on system where it converted**
  - Exit 51 to force SYSAFF to current member when leaving conversion
  - Use in conjunction with code to convert where SCHENV available
- **Sample code to insert a card in exit 4/54**
  - /*ADDCARD statement (OK, not really useful functionally)
- **Exit 0 converted**
  - Dynamic tables
  - Name/Token pair to point to UCT
  - HASX00B is loadable as $HASP.EXIT0

Migration experiences

- **Not a lot of experience yet, but some of the things we have seen:**
  - PCEID is not available to determine type of processor for specific processing
    - Use ***xxx*DEVTP** field from JRW/SRW/JTW/STW – or use generic name NJEDEVTP
  - Both exits in the “pairs” are not necessarily needed
    - Old exit 20 was used by internal reader only (test for PCEID=PCEINRID)
    - Only exit 50 was needed as a replacement (exit 20 not called for internal readers)
  - SAF checks do not need to be invoked via $SUBIT from USER environment exits
    - May require dual path if main task also needs to do check
    - $SEAS calls look the same but expand differently based on environment
  - Lots of different techniques to implement/package a common exit
  - Source mods – the code is all very different now.
  - $ENVIRON – can be a challenge getting it right – use sample exits HASXJECL and others as a model

- **Session 2665, JES2 z/OS 1.7 Exit Migration User Experiences, Thursday 3:00**

JES2 and Rel 7 Migration Offering (fee)

Have an IT specialist update your exits

US contact is **John Hock/Phoenix/IBM**
IBM Global Services
Certified Consulting IT Specialist
zSeries Delivery Team
Phone: 602-217-2206,
Elsewhere, ask Chip Wood