What’s New in z/OS Language Environment?

Thomas Petrolino
IBM Poughkeepsie
tapetro@us.ibm.com

SHARE in Orlando
August, 2011
Session 9650
Trademarks

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

- CICS®
- IMS
- Language Environment®
- OS/390®
- z/OS®

* Registered trademarks of IBM Corporation

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.
IEEE is a trademark in the United States and other countries of the Institute of Electrical and Electronics Engineers, Inc.
POSIX® is a registered Trademark of The IEEE.
SET and Secure Electronic Transaction are trademarks owned by SET Secure Electronic Transaction LLC.
Other company, product, or service names may be trademarks or service marks of others.

* 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 multi-programming 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.
Agenda

- New Function APARs
- What’s New in z/OS V1.13?
- What’s New in z/OS V1.12?

- Additional information available in Appendix:
  - What’s New in z/OS V1.11?
  - Sources for Additional Information
New Function APARS

- CEEPIPI call_sub with writable static
New Function APARs

- PK99010 – CEEPIPI call_sub with writable static
  - V1R9: UK52873  V1R10: UK52874  V1R11: UK52875
  - CEEPIPI call_sub of a subroutine with writable static where C/C++ is not the language of the entry point
    - Supports CEEFETCH and CEEPGRID as well
  - NOTE: PE – PM27753
    - Affects AMODE 24 programs statically linked with C
    - C is inherently 31bit, so code change returned function descriptor in 31bit storage that AMODE 24 program could not access.
    - Don’t need to be calling the C program to be affected.
What’s new in z/OS R13?

- Removal of RTO Usermods
- High Register Support
- CEEPIPI Multiple Main support
- Deferred Debug
- DCE removal
- I/O Abend recovery
- BSAM >64K Tracks
Removal of RTO Usermods

Statement of direction from the V1R13 Announcement:

*z/OS V1.13 is planned to be the last release to support changing the default Language Environment runtime options settings using SMP/E-installable USERMODs. IBM recommends using the CEEPRMxx PARMLIB member to set these options.*
Removal of RTO Usermods…

- If you are using the CEEWDOPT, CEEWCOPT or CEEWQDOP ++USERMOD sample jobs to set your installation default run-time options, start using CEEPRMxx NOW.

- If you are using the CEEWDOPT, CEEWCOPT or CEEWQDOP ++USERMOD sample jobs and “cloning” copies of Language Environment modules, see the Hot Topics article, “CEEROPT and the Attack of the Clones” (issue #19, P95)
  
  http://www-03.ibm.com/systems/z/os/zos/bkserv/hot_topics.html

- If anyone still believes they need these CSECTs or ++USERMODs please see me!
High Register Support

- PL/I and C/C++ compilers are capable of exploiting high halves of 64-bit general purpose registers in AMODE 31 applications
  - Integer arithmetic operations
  - Instructions introduced by the High Word Facility
- Language Environment has been enhanced:
  - to ensure this usage does not impact callers
  - to provide full register contents for diagnostic purposes
High Register Support

- Support to save/restore full 64-bit register contents when AMODE 31 application is called from the operating system or by another driver program
- AMODE 31 CEEDUMP support to display full 64 bit registers
  - When unavailable the high half of the 64 bit register is displayed as ‘******
- AMODE 31 LEDATA support to display high halves of 64 bit registers when formatting the MCH control block
- CEEDUMP/LEDATA support rolled back via PM04026
  - V1R10: UK59090  V1R11: UK59091
High Register Support

- PM04026 – High Register Support
  - CEEDUMP

Machine State:

ILC..... 0002   Interruption Code..... 0009
PSW..... 078D2400 A19C60FE
GPR0..... 00000000_00000000   GPR1..... 00000000_0000000A   GPR2.....
00000000_A1CD09BC   GPR3..... 00000000_219C60B8
GPR4..... 00000000_2199D2D8   GPR5..... 00000000_21F91A00   GPR6.....
00000000_21F92AC8   GPR7..... 00000000_219BDE40
GPR8..... 00000000_A19C63A8   GPR9..... 00000000_21F93368   GPR10....
00000000_A19C6070   GPR11.... 00000000_A19C60A0
GPR12.... 00000000_21713B58   GPR13.... 00000000_2199D6D8   GPR14....
00000000_00000000   GPR15.... 00000000_00000006
High Register Support

- **PM04026 – High Register Support**
  - **IPCS**

  Machine State
  
  +000248  MCH_EYE:ZMCH
  +000250  GPR00:00000000  GPR01:0000000A
  +000258  GPR02:A1CD09BC  GPR03:219C60B8
  +000260  GPR04:2199D2D8  GPR05:21F91A00
  +000268  GPR06:21F92AC8  GPR07:219BDE40
  +000270  GPR08:A19C63A8  GPR09:21F93368
  +000278  GPR10:A19C6070  GPR11:A19C60A0
  +000280  GPR12:21713B58  GPR13:2199D6D8
  +000288  GPR14:00000000  GPR15:00000006
  +000290  PSW:078D2400  A19C60FE
## High Register Support

- **PM04026 – High Register Support**
  - **IPCS**

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>+000388</td>
<td>GPR_H00:00000000</td>
<td>GPR_H01:00000000</td>
</tr>
<tr>
<td>+000390</td>
<td>GPR_H02:00000000</td>
<td>GPR_H03:00000000</td>
</tr>
<tr>
<td>+000398</td>
<td>GPR_H04:00000000</td>
<td>GPR_H05:00000000</td>
</tr>
<tr>
<td>+0003A0</td>
<td>GPR_H06:00000000</td>
<td>GPR_H07:00000000</td>
</tr>
<tr>
<td>+0003A8</td>
<td>GPR_H08:00000000</td>
<td>GPR_H09:00000000</td>
</tr>
<tr>
<td>+0003B0</td>
<td>GPR_H10:00000000</td>
<td>GPR_H11:00000000</td>
</tr>
<tr>
<td>+0003B8</td>
<td>GPR_H12:00000000</td>
<td>GPR_H13:00000000</td>
</tr>
<tr>
<td>+0003C0</td>
<td>GPR_H14:00000000</td>
<td>GPR_H15:00000000</td>
</tr>
</tbody>
</table>
CEEPIPI Multiple Main Support

- Additional interfaces provided in Preinit to facilitate conversion from the Preinitialization Compatibility Interface (PICI) to Preinit / CEEPIPI
  - Support for multiple main environments on one TCB;
  - Support for a user word that can be accessed from both outside and within a Preinit environment
CEEPIPI Multiple Main Support

Support for Multiple Main Environments on one TCB

- New CEEPIPI function: init_main_dp
- Allows the Preinit assembler driver to create multiple main CEEPIPI environments on the same TCB
- Main programs can be called on these environments, but only one call can be active at a time on a given TCB
CEEPIPI Multiple Main Support

Support for Multiple Main Environments on one TCB...

CALL CEEPIPI(init_main_dp,ceexptbl_addr,service_rtns,token)

- init_main_dp (input) - A fullword containing the init_main_dp function code (integer value = 19).
- ceexptbl_addr (input) - A fullword containing the address of the PreInit table to be used during initialization of the new environment.
- service_rtns (input) - A fullword containing the address of the service routine vector or 0, if there is no service routine vector.
- token (output) - A fullword containing a unique value used to represent the environment.
CEEPIPI Multiple Main Support

Support for Preinit User Word

- Facilitates communication between the Preinit assembler driver and the user code running within a Preinit environment

- Preinit assembler driver uses CEEPIPI interfaces to access the user word
  - CEEPIPI(set_user_word,...) sets the user word value
  - CEEPIPI(get_user_word,...) retrieves the user word value from the last set_user_word call
CEEPIPI Multiple Main Support

Support for Preinit User Word...

- Code running within Preinit environment accesses the user word from within the CAA control block
  - Field CEECAA_USER_WORD in the assembler CEECAA mapping
    - 4 byte field located at offset +3F0x
  - Modifications to this field by the user code running in the Preinit environment are not saved between CEEPIPI calls
    - Next CEEPIPI call will use value from last set_user_word call
CEEPIPI Multiple Main Support

Support for Preinit User Word…

CALL CEEPIPI(set_user_word,token,value)

- **set_user_word** (input) - A fullword containing the set_user_word function code (integer value = 17)
- **token** (input) - A fullword with the value of the token of the environment
- **value** (input) - A fullword value that will be used to initialize the user word in the initial thread CAA when the application is invoked
CEEPIPI Multiple Main Support

Support for Preinit User Word…

CALL CEEPIPI(get_user_word,token,value)

- get_user_word (input) - A fullword containing the get_user_word function code (integer value = 18)
- token (input) - A fullword with the value of the token of the environment
- value (output) - A fullword that will be returned containing the current value that will be used to initialize the CAA user word when the next application is invoked
Deferred Debug Support

- Support to allow debugging to start at a particular C/C++ entry point
  - Currently available for COBOL and PL/I
- New callable service CEEKRGPM provided
- New RCB fields
  - CEERCB_PMUSER – pm_user address supplied on the CEEKRGPM call
  - CEERCB_PMADDR – pm_addr address supplied on the CEEKRGPM call
- Support available via PTFs for APAR PM15192
  - V1R10: UK90027  V1R11: UK90028  V1R12: UK90029
Deferred Debug Support

- CEEKRGPM callable service

```c
--- Syntax -----------------------------------------------

<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CEEKRGPM ( pm_addr, rsvd_word1, pm_user, [fc] )</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>POINTER     *pm_addr;</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>INT4        *rsvd_word1;</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>POINTER     *pm_user;</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>FEED_BACK   *fc;</td>
</tr>
</tbody>
</table>

-----------------------------------------------
```

- Call this CWI interface as follows:

```
L     R15,CEECAACELV-CEECAA(R12)
L     R15,68(R15)
BALR  R14,R15
```
Deferred Debug Support

- CEEKRGPM callable service
  - pm_addr (input)
    - The address of a pattern match routine that is to be registered or zero when no pattern match routine should be registered (de-registration).
  - rsvd_word1 (input)
    - A fullword reserved for future use. This must be set to zero.
  - pm_user (input)
    - The address of an area supplied by the user for the user’s use, or zero when no user area is provided. (The user is responsible for freeing this area, if necessary.)
  - fc (output/optional)
Deferred Debug Support

- **CEEPIPI call_sub**
  - If CEERCB_PMADDR is non-zero
    - The pattern match routine will be called specifying function code 177, along with the name and entry point (from the PIPI table) of the call_sub routine about to be invoked, as well as the supplied pm_user address.
    - This call will occur just prior to calling the debug event handler with PIPI Subroutine Initialization event (115)
    - The pattern match routine will not be called for call_sub_addr, call_sub_addr_nochk or call_sub_addr_nochk2 requests
    - The pattern match routine will not be called if no routine name is available in the PIPI table
DCE Removal

- Starting in z/OS V1R13, the z/OS Distributed Computing Environment (DCE) and Distributed Computing Environment Security Server (DCE Security Server) will no longer ship with z/OS.
- Minor updates made to documentation and C headers
DCE Removal

- pthread.h
  - Removed inclusion of `<dce/dce_pthreads.h>`
  - Added a `#error` message in place of the header inclusion to gracefully fail the compile with a meaningful message.
  - This header also contains equates `__COND_DCE` and `__MUTEX_DCE`. There are no references in the external publications for these definitions. Definitions for these have been left in the header for compatibility.
DCE Removal

- signal.h
  - Removed inclusion of `<dce/dce_signal.h>`
  - Added a `#error` message in place of the header inclusion to gracefully fail the compile with a meaningful message.
  - This header also contains a signal definition called SIGDCE. References to this signal have been removed or reworded in the external publications, but signal definition in the header has been left for compatibility.
DCE Removal

- Update CEE5224W explanation

CEE5224W The signal SIGDCE was received.

Explanation: **DCE has been removed as of z/OS V1R13.** On previous releases the SIGDCE signal was generated as a result of a MODIFY DCEKERN,DEBUG pid= command. It communicates to a DCE-enabled process a desire to enable DCE run-time debug messages. If the target process is not a DCE process, the target process does not know how to handle SIGDCE.

Programmer response: None.

System action: No system action taken.

Symbolic feedback code: CEE538
I/O Abend Recovery

- The C-RTL is unable to ignore certain DFSMS abends, requiring application developers to write condition handlers or SIGABND handlers to attempt recovery.

- I/O Abend Recovery provides a mechanism to enable the C-RTL to recover gracefully from an abend condition during output or CLOSE processing, when the abend cannot be ignored.

- The C-RTL function that triggered the abend condition will gracefully return to the application instead of bringing down the enclave (if condition handling is not in effect).
I/O Abend Recovery

Two different ways to invoke abend recovery behavior:

- New fopen()/freopen() keyword
- New environment variable

Either method can be used to control how the C-RTL treats abend conditions that cannot be ignored.

When either method is set up to recover from the abend, the C-RTL will instruct the function to return a failing value to the application and set errno to 92.

Diagnostic information will also be set in the __amrc structure.
I/O Abend Recovery

- `fopen() / freopen()` keyword usage
  - `abend=abend | recover`
  - `abend` instructs the runtime library to ignore abend conditions that can be ignored. No attempt is made to recover from abend conditions that cannot be ignored.
  - `recover` instructs the runtime library to attempt to recover from an abend issued during certain low-level I/O operations (WRITE / CHECK sequence and CLOSE).
  - This method specifies the behavior for only the stream being opened.
  - This method overrides the setting of the `_EDC_IO_ABEND` environment variable

```c
fopen("//'myfile.data'","wb,type=record,abend=recover");
```
**I/O Abend Recovery**

- **Environment variable usage**
  - Environment variable name is `_EDC_IO_ABEND`
  - **ABEND** and **RECOVER** are the possible values
    - The values invoke the same behavior as their relative fopen() keyword values.
    - When unset or set to something other than RECOVER, the default behavior is ABEND.
    - The setting of the environment variable defines the behavior for the life of an open stream
    - The environment variable can be overridden by the fopen() keyword.

```c
setenv("_EDC_IO_ABEND", "RECOVER", 1);
```
I/O Abend Recovery

- **CEECAAA Updates (with offsets)**
  - **CEECAASHAB_RECOVER_IN_ESTAE_MODE (+30C)**
    - [Bit in the CEECAAAFLAG1 field] When ON, the Language Environment ESTAE resumes to the abend shunt in the mode and key in which the Language Environment ESTAE was established.
  - **CEECAASHAB_KEY (+30D)**
    - [Character] IPK result when CEECAASHAB is set.
  - These fields are added to safe guard against key switching and doing a retry in a different key than which the recovery routine was established.
BSAM Greater Than 64K Tracks

- XL C/C++ Run-time Library support for large format sequential data sets greater than 65,535 tracks/volume when opened for BSAM (seek) under binary and text I/O
  - This is Part III of the C/C++ RTL efforts to exploit DFSMSdfp support for large format sequential data sets
    - V1R8 – data sets opened for QSAM I/O
    - V1R12 – data sets opened for BSAM (seek) under record I/O
  - Note: Allocation of a new large format sequential data set can be accomplished by specifying the keyword DSNTYPE=LARGE on a JCL DD statement or using the dynamic allocation equivalent.
  - Supported data set types: large format sequential data sets that are single or multivolume, residing on SMS-managed or non-SMS managed devices, catalogued or uncatalogued.
BSAM Greater Than 64K Tracks

- XL C/C++ Run-time Library support for large format sequential data sets...
  - Invoked by calling the fopen() function on a pre-existing large format sequential data set (DNSTYPE=LARGE was specified).
  - New macro __DSNT_LARGE added for use with dynalloca() function.
  - No support for the fopen() or freopen() equivalent of DSNTYPE=LARGE
  - Once the data set is opened, other OS I/O functions can be used to process the stream.
BSAM Greater Than 64K Tracks

- Large Files versions of ftello() and fseeko() have been added that work on large format sequential data sets that are opened for any type of I/O (record, binary or text)
  - Allows reporting of and repositioning, either directly or relatively, to offsets greater than 2GB – 1.
    - AMODE 31 ftell() and fseek() behavior remains unchanged: offsets used for reporting and repositioning are still limited to 2GB - 1.
  - In order to use the large files versions of ftello() and fseeko(), define the following feature test macro in your application:
    
    ```
    #define _LARGE_FILES 1
    ```
  - fgetpos() and fsetpos() can be used with large files without any source code modification
What’s new in z/OS R12?

- CEEPRMxx OVR/NONOVIR support
- BAM XTIOT support
- Heap Storage Reallocation Performance
CEEPRMxx OVR/NONOVR Support

- CEEPRMxx Override/Nonoverride support
  - Existing syntax will be unchanged and fully supported (no migration action)
  - New syntax will match current CEEDOPT usermod syntax
    - ALL31(ON) – existing
    - ALL31=((ON),OVR) – new
  - This includes “NOxxxxx” options
    - NODEBUG – existing
    - DEBUG=((OFF),OVR) – new
    - Suboption is required for these “NO” options
CEEPRMxx OVR/NONOVR Support

- CEEPRMxx Override/Nonoverride support
  - Will work in SETCEE, SET CEE and syntax checker.
  - D CEE updated to show non-overrideable when appropriate

CEE=(A8)
LAST WHERE SET OPTION
---------------------------------------------------------
SETCEE Non-overrideable ALL31(ON)
CEEPRMA8 Non-overrideable RPTOPTS(ON)
CEEPRMA8 Non-overrideable TRAP(ON,SPIE)
CEEPRMxx OVR/NONOVR Support

- CEEPRMxx Override/Nonoverride support
  - SETCEE CEEDOPT,TRAP=((ON),OVR)
    - Acceptable syntax in CEEPRMxx
      - leaving out the 2nd suboption
  - D CEE updated to show only specified suboptions

<table>
<thead>
<tr>
<th>CEE=(A8)</th>
<th>LAST WHERE SET</th>
<th>OPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>SETCEE Non-overrideable</td>
<td>ALL31(ON)</td>
<td></td>
</tr>
<tr>
<td>CEEPRMA8 Non-overrideable</td>
<td>RPTOPTS(ON)</td>
<td></td>
</tr>
<tr>
<td>SETCEE Non-overrideable</td>
<td>TRAP(ON,)</td>
<td></td>
</tr>
</tbody>
</table>
What's New in z/OS V1.12?

- CEA0107I COMMON EVENT ADAPTER IS RUNNING IN FULL FUNCTION MODE.
- SET CEE=J7
  CEE3742I THE SET CEE COMMAND HAS COMPLETED.
- SETCEE CEE=CEEDOPT,ALL31=((OFF),NONOVR),CHECK=((ON),OVR),DEBUG=((OFF),NONOVR),
  PRTUNIT(8)
  CEE3743I THE SETCEE COMMAND HAS COMPLETED.
- d cee,ceedopt
  CEE3745I 07.34.19 DISPLAY CEE=CEEDOPT
  CEE=(J7)

---

<table>
<thead>
<tr>
<th>LAST WHERE SET</th>
<th>OPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>-----------------</td>
<td>--------</td>
</tr>
<tr>
<td>PARMLIB(CEEPRM)</td>
<td>ABPERC(NONE)</td>
</tr>
<tr>
<td>SETCEE Non-overrideable</td>
<td>ALL31(OFF)</td>
</tr>
<tr>
<td>PARMLIB(CEEPRM)</td>
<td>ANYHEAP(16384,,FREE)</td>
</tr>
<tr>
<td>CEEPRM Non-overrideable</td>
<td>BELOWHEAP(16384,18192,FREE)</td>
</tr>
<tr>
<td>SETCEE command</td>
<td>CHECK(ON)</td>
</tr>
<tr>
<td>SETCEE Non-overrideable</td>
<td>NODEBUG</td>
</tr>
<tr>
<td>PARMLIB(CEEPRM)</td>
<td>FILETAG(NOAUTOVT,NOAUTOTAG)</td>
</tr>
<tr>
<td>CEEPRM Non-overrideable</td>
<td>HEAP(32768,32768,ANYWHERE,KEEP,18192,4096)</td>
</tr>
<tr>
<td></td>
<td>HEAPCHK(OFF,1,0,0,0,1024,0,1024,0)</td>
</tr>
<tr>
<td></td>
<td>LIBSTACK(4096,4096,FREE)</td>
</tr>
<tr>
<td></td>
<td>PRTUNIT(8)</td>
</tr>
<tr>
<td></td>
<td>RECPAD(0)</td>
</tr>
<tr>
<td></td>
<td>RPTOPTS(ON)</td>
</tr>
<tr>
<td></td>
<td>TRACE(OFF,4096,DUMP,LE=0)</td>
</tr>
<tr>
<td></td>
<td>TRAP(ON,SPIE)</td>
</tr>
</tbody>
</table>

IEE612I CN=POSIXCON DEVMUN=03E0 SYS=SY1

---

IEE163I MODE= RD
BAM XTIOT Support

- Language Environment now supports the use of certain DDNAMEs that have been dynamically allocated with XTIOT, UCB nocapture, or DSAB-above-the-line options specified in the SVC99 parameters (S99TIOEX, S99ACUCB, S99DSABA flags).
  - CEEDUMP DDNAME – supported
  - CEEOPTS DDNAME – NOT supported
  - MSGFILE DDNAME – supported

- C/C++ function fopen() and dynalloc() updated to support the use of new XTIOT options
Heap storage reallocation

- The Language Environment callable service CEECZST (and the C/C++ function realloc()) support a new environment variable
  - _CEE_REALLOC_CONTROL
    - Parameter 1 – Lower bound threshold
      - The number of bytes above which the tolerance percentage (parm 2) will be applied
    - Parameter 2 – Tolerance Percentage
      - The percentage of extra storage to be obtained
      - 0 to 100
Heap storage reallocation

- Example
  - `_CEE_REALLOC_CONTROL=100,20`
    - First request is for 80 bytes
      - Storage obtained as normal
    - A request to change this storage to 90 bytes
      - Storage obtained as normal
    - A request to change this storage to 100 bytes
      - At or above threshold, percentage is applied
      - Storage obtained is 120 bytes (100 + 100 * 20%)
    - A request to change this storage to 110 bytes
      - No storage need be obtained (we already have 120 bytes)
Heap storage reallocation

- Can be very useful for programs that make many requests to reallocate storage larger than originally requested.
  - Many string manipulation routines make heavy use of storage reallocation.
- If tolerance percentage is 0 or _CEE_REALLOC.CONTROL is not set no change in behavior.
The End..

Thank you!
Appendix

- What’s New in z/OS V1.11?
- Sources for Additional Information
What’s new in z/OS R11?

- Assembler Macro Updates
- CICS Additional Floating Point Support
- CELQPIPI service routines update
- Additional diagnostics for HEAPPOOLS
Assembler Macro Updates

- Create CEEGLOB assembler macro similar to IBM Language Environment for z/VSE
- Add support in CEEPPA for the SERVICE keyword option
- Add support in CEEENTRY for the RMODE and AMODE keyword options
- Add support in CEEFETCH to handle both Language Environment and non-Language Environment code and provide support to do an “Language Environment-load” if module previously loaded
Assembler Macro Updates

CEEGLOB global assembler variables:

- &CEEGPRO (alias &GPRO) Product number
- &CEEGVER (alias &GVER) Product version
- &CEEGREL (alias &GREL) Product release
- &CEEGMOD (alias &GMOD) Product modification level
- &CEEGENV (alias &GENV) OS environment from which the macro has been invoked
Assembler Macro Updates

CEEPPPA Service Keyword

- New SERVICE keyword to set the service level string for a routine.
  - Syntax: SERVICE=service_string

- The service string length and contents are located following the timestamp and version information.

- This field is not interrogated by Language Environment.

- The SERVICE keyword can only be specified on the first CEEPPPA macro in the assembler source, all other instances of the keyword are ignored.

- When the SERVICE keyword is in use, the timestamp is generated automatically, the TSTAMP option is forced to YES even when the user specified TSTAMP=NO.
  - If the TSTAMP option if forced to YES the following severity 4 MNOTE is generated:
    SERVICE PARAMETER SPECIFIED TSTAMP PARAMETER FORCED TO 'YES'
Assembler Macro Updates

CEEPPA Service Keyword

192+* Time Stamp
193+*, Time Stamp = 2009/02/02 15:16:00 01-CEEPP
194+*, Version 1 Release 1 Modification 0 01-CEEPP
195+CEETIMES DS 0F 01-CEEPP
196+ DC CL4'2009' Year 01-CEEPP
197+ DC CL2'02' Month 01-CEEPP
198+ DC CL2'02' Day 01-CEEPP
199+ DC CL2'15' Hours 01-CEEPP
200+ DC CL2'16' Minutes 01-CEEPP
201+ DC CL2'00' Seconds 01-CEEPP
202+ DC CL2'1' Version 01-CEEPP
203+ DC CL2'1' Release 01-CEEPP
204+ DC CL2'0' Modification 01-CEEPP
205+ DC AL2(6) Length of Service String @D2A 01-CEEPP
206+ DC C'011100' Service parm @D2A 01-CEEPP
Assembler Macro Updates

Example with CEEGLOB and CEEPPA

```
GBLC &GVER,&GREL,&GMOD
CEEGLOB
ASMTSTRC CEEENTRY PPA=MYPAA,BASE=R11,MAIN=YES
   LA 3,12
   ST 3,RETCODE
   LA 2,8
   LA 3,0
   ST 2,(0,3)
   CEETERM RC=RETCODE,MODIFIER=0
RETCODE DS F
R3 EQU 3
R11 EQU 11
LTORG ,
* The service level string is set to the concatenation of the CEEGLOB values for
* the Version, Release and Modification Level
MYPAA CEEPPA SERVICE=&GVER.&GREL.&GMOD
CEEDSA ,
CEECAA ,
CEEOCB ,
END ASMTSTRC
```
Assembler Macro Updates

Sample CEEDUMP output

Traceback:

<table>
<thead>
<tr>
<th>DSA</th>
<th>Entry</th>
<th>E Offset</th>
<th>Statement</th>
<th>Load Mod</th>
<th>Program Unit</th>
<th>Service</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CEEHDP</td>
<td>+00004B34</td>
<td>CEEPLPKA</td>
<td>CEEHDP</td>
<td>HLE7750</td>
<td>Call</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>ASMTSTRC</td>
<td>+0000008A</td>
<td>ASMRC01G</td>
<td>ASMTSTRC</td>
<td>011100</td>
<td>Exception</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DSA</th>
<th>DSA Addr</th>
<th>E Addr</th>
<th>PU Addr</th>
<th>PU Offset</th>
<th>Comp Date</th>
<th>Compile Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2159C0B0</td>
<td>0D1BB3E0</td>
<td>0D1BB3E0</td>
<td>+00004B34</td>
<td>20080319</td>
<td>CEL</td>
</tr>
<tr>
<td>2</td>
<td>2159C030</td>
<td>0006D000</td>
<td>0006D000</td>
<td>+0000008A</td>
<td>20080512</td>
<td>ASM</td>
</tr>
</tbody>
</table>
Assembler Macro Updates

CEEENTRY updated with RMODE and AMODE keyword

- New RMODE and AMODE keywords that will allow for the specification of the modules CSECT RMODE and AMODE settings. The default for both will remain ANY.

- Syntax:
  
  \[
  \text{RMODE=} \text{<ANY | 24 | 31> the default, if unspecified, is ANY} \\
  \text{AMODE=} \text{<ANY | 24 | 31 | ANY31> the default, if unspecified, is ANY}
  \]

- Example:
  
  \[
  \text{MAIN CEEENTRY PPA=MAINPPA, \ldots, RMODE=24, AMODE=31}
  \]
Assembler Macro Updates

CEEFETCH Enhancements

- Three new keywords are introduced in CEEFETCH: FTCHINFO, ENTRYPNT, and SCOPE=PROCESS

Syntax

```
SCOPE=ENCLAVE
>>__label__CEEFETCH__ _______________________ ,,.|__________________________|___>
|___NAME=__name___________|___SCOPE=THREAD___|
|___NAMEADDR=__nameaddr__|___SCOPE=PROCESS___|
|___ENTRYPNT=__entrypt___|

>_ _______________________ ,,.|.__________________________<<
|___FTCHINFO=__ftchinfo___|
```

What's New in z/OS V1.11?
Assembler Macro Updates

CEEFETCH Enhancements

**SCOPE=PROCESS**

- Indicates that the load is to be scoped to the process level. Modules loaded at the process level are deleted automatically at process termination.
- SCOPE=ENCLAVE remains the default
- SCOPE=THREAD is still supported
Assembler Macro Updates

CEEFETCH Enhancements

**FTCHINFO=**__ftchinfo

- Used in combination with NAME or NAMEADDR to request a *load attempt* on a target module whose characteristics are unknown.
- Set to a previously allocated storage area in the form of a register (enclosed in parentheses) or the name of a fullword address variable, that will contain any information discovered about the target module, see CEEFTCH for mapping details.
- If the module is identified as a Language Environment conforming AMODE 24 or AMODE 31 subroutine, then processing would be as normal (added to the member list, function pointer obtained, added to the load list table), otherwise only a load of the target will be attempted.
Assembler Macro Updates

CEEFETCH Enhancements

ENTRYPT=ENTRYPT

- Used in combination with FTCHINFO to obtain information about a previously loaded module and to do any corresponding processing on it as if it was initially loaded by CEEFETCH.

- The NAME and NAMEADDR keywords are mutually exclusive with ENTRYPT.

- If the module is identified as a Language Environment conforming AMODE 24 or AMODE 31 subroutine, then it will be added to the member list, have a function pointer obtained, and added as an entry in to the load list table.

- Set to the entry point for a previously loaded target module stored either in the form of a register (enclosed in parentheses) or the name of a fullword address variable.
Assembler Macro Updates

CEEFETCH Enhancements

- New messages/feedback codes associated with CEEFETCH

<table>
<thead>
<tr>
<th>Symbolic Feedback</th>
<th>Severity</th>
<th>Message Number</th>
<th>Message Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CEE3DV</td>
<td>3</td>
<td>3519</td>
<td>The version specified in the CEEFTCH control block passed to the CEEFETCH macro is not supported.</td>
</tr>
<tr>
<td>CEE3QS</td>
<td>1</td>
<td>3932</td>
<td>The system service CSVQUERY failed with return code &lt;return_code&gt; and reason code 0.</td>
</tr>
</tbody>
</table>
Assembler Macro Updates

CEEFTCH

- macro used to generate a mapping for the module information in the FTCHINFO storage area

___ Syntax ________________________________

| >>__CEEFTCH__ ________________ ______>< |

| >>__CEEFTCH__ ________________ ______>< |

| _DSECT=_YES_ |

| _DSECT=_No__ |

| _______________________________________ |
Assembler Macro Updates

CEEFTCH

- **DSECT=YES**
  - Indicates that a DSECT mapping should be generated.
  - This is the default for the mapping if the DSECT option is not specified.

- **DSECT=NO**
  - Indicates that a data area mapping should be generated.
  - The following tables show the format of the CEEFTCH mapping Version 1 (CEEFTCH_VERSION = 1).
## CEEFTCH mapping

<table>
<thead>
<tr>
<th>Offset Dec</th>
<th>Offset Hex</th>
<th>Type</th>
<th>Len</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(0)</td>
<td>Structure</td>
<td>64</td>
<td>CEEFTCH</td>
<td>Start of CEEFETCH</td>
</tr>
<tr>
<td>0</td>
<td>(0)</td>
<td>Character</td>
<td>8</td>
<td>CEEFTCH_EYE_CATCHER</td>
<td>Eyecatcher</td>
</tr>
<tr>
<td>8</td>
<td>(8)</td>
<td>Unsigned</td>
<td>2</td>
<td>CEEFTCH_VERSION</td>
<td>Version requested</td>
</tr>
<tr>
<td>10</td>
<td>(A)</td>
<td>BIT(8)</td>
<td>1</td>
<td>CEEFTCH_FLAGS1</td>
<td>CEEFETCH flags1</td>
</tr>
<tr>
<td>10</td>
<td>(A)</td>
<td>BIT(1)</td>
<td>1</td>
<td>CEEFTCH_A24</td>
<td>X’80’ target is AMODE 24</td>
</tr>
<tr>
<td>10</td>
<td>(A)</td>
<td>BIT(1) POS(2)</td>
<td>1</td>
<td>CEEFTCH_A31</td>
<td>X’40’ target is AMODE 31</td>
</tr>
<tr>
<td>10</td>
<td>(A)</td>
<td>BIT(1) POS(3)</td>
<td>1</td>
<td>CEEFTCH_A64</td>
<td>X’20’ target is AMODE 64</td>
</tr>
<tr>
<td>10</td>
<td>(A)</td>
<td>BIT(1) POS(4)</td>
<td>1</td>
<td>CEEFTCH_XPLINK</td>
<td>X’10’ target is XPLINK</td>
</tr>
<tr>
<td>10</td>
<td>(A)</td>
<td>BIT(1) POS(5)</td>
<td>1</td>
<td>CEEFTCH_LE</td>
<td>X’08’ target is Language Environment conforming</td>
</tr>
<tr>
<td>10</td>
<td>(A)</td>
<td>BIT(1) POS(6)</td>
<td>1</td>
<td>CEEFTCH_MAIN</td>
<td>X’04’ target is MAIN</td>
</tr>
<tr>
<td>10</td>
<td>(A)</td>
<td>BIT(1) POS(7)</td>
<td>1</td>
<td>CEEFTCH_SUB</td>
<td>X’02’ target is a SUB</td>
</tr>
<tr>
<td>10</td>
<td>(A)</td>
<td>BIT(1) POS(8)</td>
<td>1</td>
<td>CEEFTCH_DLL</td>
<td>X’01’ target is DLL</td>
</tr>
</tbody>
</table>
### CEEFTCH mapping

<table>
<thead>
<tr>
<th>Offset Dec</th>
<th>Offset Hex</th>
<th>Type</th>
<th>Len</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>(B)</td>
<td>BIT(8)</td>
<td>1</td>
<td>CEEFTCH_FLAGS2</td>
<td>CEEFTCH flags2</td>
</tr>
<tr>
<td>11</td>
<td>(B)</td>
<td>BIT(1)</td>
<td>1</td>
<td>CEEFTCH_SEGMENTED</td>
<td>X’80’ target module is divided into multiple initial load segments (deferred load segments, if any, are not counted)</td>
</tr>
<tr>
<td>11</td>
<td>(B)</td>
<td>BIT(1) POS(2)</td>
<td>1</td>
<td>CEEFTCH_CICS</td>
<td>X’40’ CICS environment</td>
</tr>
<tr>
<td>11</td>
<td>(B)</td>
<td>BIT(6) POS(3)</td>
<td>1</td>
<td>*</td>
<td>Available</td>
</tr>
<tr>
<td>12</td>
<td>(C)</td>
<td>SIGNED</td>
<td>4</td>
<td>*</td>
<td>Available</td>
</tr>
<tr>
<td>16</td>
<td>(10)</td>
<td>ADDRESS</td>
<td>8</td>
<td>CEEFTCH_CEESTART64</td>
<td>Address of 64bit CEESTART</td>
</tr>
<tr>
<td>16</td>
<td>(10)</td>
<td>SIGNED</td>
<td>4</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>(14)</td>
<td>ADDRESS</td>
<td>4</td>
<td>CEEFTCH_CEESTART</td>
<td>Address of 31bit CEESTART</td>
</tr>
</tbody>
</table>
### CEEFTCH mapping

<table>
<thead>
<tr>
<th>Offset Dec</th>
<th>Offset Hex</th>
<th>Type</th>
<th>Len</th>
<th>Name (Dim)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>(18)</td>
<td>ADDRESS</td>
<td>8</td>
<td>CEEFTCH_MOD64</td>
<td>Address of 64bit target</td>
</tr>
<tr>
<td>24</td>
<td>(18)</td>
<td>SIGNED</td>
<td>4</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>(1C)</td>
<td>ADDRESS</td>
<td>4</td>
<td>CEEFTCH_MOD</td>
<td>Address of 31bit target</td>
</tr>
<tr>
<td>32</td>
<td>(20)</td>
<td>SIGNED</td>
<td>8</td>
<td>CEEFTCH_MOD_LEN64</td>
<td>Length of 64bit target</td>
</tr>
<tr>
<td>32</td>
<td>(20)</td>
<td>SIGNED</td>
<td>4</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>(24)</td>
<td>SIGNED</td>
<td>4</td>
<td>CEEFTCH_MOD_LEN</td>
<td>Length of 31bit target</td>
</tr>
<tr>
<td>40</td>
<td>(28)</td>
<td>ADDRESS</td>
<td>8</td>
<td>CEEFTCH_EP64</td>
<td>Address of 64bit EntryPt</td>
</tr>
<tr>
<td>40</td>
<td>(28)</td>
<td>SIGNED</td>
<td>4</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>(2C)</td>
<td>ADDRESS</td>
<td>4</td>
<td>CEEFTCH_EP</td>
<td>Address of 31bit EntryPt</td>
</tr>
<tr>
<td>48</td>
<td>(30)</td>
<td>UNSIGNED</td>
<td>8</td>
<td>*</td>
<td>Available</td>
</tr>
<tr>
<td>56</td>
<td>(38)</td>
<td>UNSIGNED</td>
<td>8</td>
<td>*</td>
<td>Available</td>
</tr>
</tbody>
</table>
ASMFT3E1 example

Example using FTCHINFO to load a module and test the mapping bits to determine characteristics:

```assembly
*=====================================================================
* USE NEW FTCHINFO SUPPORT IN CEEFETCH TO ATTEMPT A LOAD
* OF TARGET MODULE 31BIT 'CPPSUBRT'
*=====================================================================

ASMFT3E1 CEEENTRY PPA=MYPPA,MAIN=YES,BASE=4,AUTO=WORKSIZE,
    X
    ENCLAVE=YES
    USING WORKAREA,13
    LA 2,1
    STH 2,CEEFETCH_VERSION
        SET MAP VERSION TO 1
    LA 2,CEEFETCH
        STORE ADDR OF
    ST 2,INFOPT
        CEEFETCH IN INFOPT
    CEEFETCH NAME=CPPSUBRT,
        X
        TOKEN=TOKEN1,FEEDBACK=FB2,
        X
        MF=(E,LABEL1),FTCHINFO=INFOPT,SCOPE=PROCESS
    CLC FB2(8),CEE000
        CHECK FEEDBACK CODE
    BE GOOD_FB
    CALL CEEMSG,(FB2,DEST,FB3)
        DISPLAY FEEDBACK
    CEETERM RC=16,MODIFIER=0
    B DONE
        LEAVE IF BAD
    GOOD_FB DS 0H

    BALR 14,15
```

What's New in z/OS V1.11?
ASMFT3E1 example

* TEST THE FLAG BITS
*

TM CEEFTCH_FLAG1, CEEFTCH_DLL
JZ XPLINK_T
CALL CEEMOUT, (DLLC, DEST, FB), VL, MF=(E, CALLMOUT)

XPLINK_T EQU *

TM CEEFTCH_FLAG1, CEEFTCH_XPLINK
JZ AMODE_T
CALL CEEMOUT, (XPC, DEST, FB), VL, MF=(E, CALLMOUT)

AMODE_T EQU *

TM CEEFTCH_FLAG1, CEEFTCH_A24
JZ AMODE_3
CALL CEEMOUT, (A24C, DEST, FB), VL, MF=(E, CALLMOUT)

AMODE_3 EQU *

TM CEEFTCH_FLAG1, CEEFTCH_A31
JZ AMODE_6
CALL CEEMOUT, (A31C, DEST, FB), VL, MF=(E, CALLMOUT)

AMODE_6 EQU *

TM CEEFTCH_FLAG1, CEEFTCH_A64
JZ LE_T
CALL CEEMOUT, (A64C, DEST, FB), VL, MF=(E, CALLMOUT)

LE_T EQU *

TM CEEFTCH_FLAG1, CEEFTCH_LE
JZ SUB_T
CALL CEEMOUT, (LEC, DEST, FB), VL, MF=(E, CALLMOUT)
ASMFT3E1 example

```
SUB_T EQU *
  TM CEEFTCH_FLAGS1, CEEFTCH_SUB
  JZ MAIN_T
  CALL CEEMOUT, (SUBC, DEST, FB), VL, MF=(E, CALLMOUT)

MAIN_T EQU *
  TM CEEFTCH_FLAGS1, CEEFTCH_MAIN
  JZ CICS_T
  CALL CEEMOUT, (MAINC, DEST, FB), VL, MF=(E, CALLMOUT)

CICS_T EQU *
  TM CEEFTCH_FLAGS2, CEEFTCH_CICS
  JZ SEG_T
  CALL CEEMOUT, (CICSC, DEST, FB), VL, MF=(E, CALLMOUT)

SEG_T EQU *
  TM CEEFTCH_FLAGS2, CEEFTCH_SEGMENTED
  B DONE
  CALL CEEMOUT, (SEGC, DEST, FB), VL, MF=(E, CALLMOUT)

DONE DS 0H
```

*DELETE LOADED ROUTINE*

```
CEERELES TOKEN=TOKEN1, FEEDBACK=FB2
CALL CEEMSG, (FB2, DEST, FB3)          DISPLAY FB
CEETERM RC=0, MODIFIER=0
```
ASMFT3E1 example

* ---------------------------------------------------------------
* CONSTANTS
* ---------------------------------------------------------------

TOKEN1 DS F
MODNAME DC CL8'CPPSUBRT'
FB3 DC CL12'FEEDBACKCODE'
FB2 DS CL12'FEEDBACKCODE'
DEST DC F'2' DESTINATION IS THE LE MESSAGE FILE
CEE000 DS 3F'0' SUCCESS FEEDBACK CODE

* 
LEC DC Y(LEEND-LESTR)
LESTR DC C'I AM LE.'
LEEND EQU *
* 
A24C DC Y(A24END-A24STR)
A24STR DC C'I AM AMODE24.'
A24END EQU *
* 
A31C DC Y(A31END-A31STR)
A31STR DC C'I AM AMODE31.'
A31END EQU *
* 
A64C DC Y(A64END-A64STR)
A64STR DC C'I AM AMODE64.'
A64END EQU *
ASMFT3E1 example

XPC       DC       Y(XPEND-XPSTR)
XPSTR     DC       C'I AM XPLINK.'
XPEND     EQU      *
           *
CICSC     DC       Y(CICSEND-CICSTR)
CICSTR    DC       C'I AM IN CICS.'
CICSEND   EQU      *
           *
MAINC     DC       Y(MAINEND-MAINSTR)
MAINSTR   DC       C'I AM A MAIN.'
MAINEND   EQU      *
           *
SUBC      DC       Y(SUBEND-SUBSTR)
SUBSTR    DC       C'I AM A SUBROUTINE.'
SUBEND    EQU      *
           *
DLLC      DC       Y(DLLEND-DLLSTR)
DLLSTR    DC       C'I AM A DLL.'
DLLEND    EQU      *
           *
SEGC      DC       Y(SEGEND-SEGSTR)
SEGSTR    DC       C'I AM SEGMENTED.'
SEGENDEQU   *
ASMFT3E1 example

* MYPPA CEEPPA , CONSTANTS DESCRIBING THE CODE BLOCK
* ================================================
* THE WORKAREA AND DSA
* ================================================

WORKAREA DSECT
   ORG    *+CEEDSASZ LEAVE SPACE FOR THE DSA FIXED PART
   FB DS    3F SPACE FOR A 12-BYTE FEEDBACK CODE
   *
   *
   CALLMOUT CALL ,(),VL,MF=L 3-ARGUMENT PARAMETER LIST

LABEL1 CEEFETCH MF=L
   CEEFETCH DSECT=NO
INFOPT DS A *
* EPPTR DS A *
* DS 0D
WORKSIZE EQU *-WORKAREA
   CEEDSA , MAPPING OF THE DYNAMIC SAVE AREA
   CEECAA , MAPPING OF THE COMMON ANCHOR AREA
   *
   *
END ASMFT3E1
CICS AFP Support

CICS AFP (Additional Floating Point) Support

- Prior to CICS TS Version 4, Language Environment was unable to fully support Binary Floating Point (BFP) and Decimal Floating Point (DFP)
  - Before this change, Language Environment did not fully support BFP or DFP operations in applications that run in a CICS environment.
    - It was possible to compile XL C/C++ and Enterprise PL/I programs with the AFP(VOLATILE) compiler option and do BFP/DFP operations, as long as the default floating point rounding mode was not altered.
    - In a CICS TS environment, certain BFP and DFP program checks would always result in a CEE3207 message.
      - The same program checks would result in CEE321X, CEE322X, and CEE323X messages in a non-CICS environment.
    - Floating point registers 1,3,5,7, and 8-15, along with the floating point control register (FPC) did not appear in CEEDUMPs or IPCS dumps, when running under CICS TS.
CICS AFP Support

CICS AFP (Additional Floating Point) Support

- With this new support, binary and decimal floating point operations are fully supported in the CICS TS Version 4 or later environment.
  - The AFP(VOLATILE) compiler option is no longer required
  - All applicable floating point registers 0-15 and the FPC register appear in dumps after program checks or ABENDs.
  - It is now possible to run many simultaneous programs in a CICS TS region that do binary or decimal floating point operations with non-default rounding modes, with no interference between the applications.
CICS AFP Support

CICS AFP (Additional Floating Point) Support

- Language Environment and CICS TS Version 4 and later will automatically activate the new CICS AFP support when the CICS environment is started.
- CEEDUMPs and formatted IPCS dumps will sometimes show additional registers after CICS program checks and ABENDs:
  - Floating point registers 0-15 (before this change only 2, 4, 6, 8 were included)
  - Floating point control register (FPC)
  - High registers (and low registers, as before)
  - Access registers
- Floating point 0C7 program checks are now mapped into the same CEE32xx messages in CICS and non-CICS environments.
CELQPIPI Enhancements

CELQPIPI Service Routines

- AMODE 64 Preinitialization (CELQPIPI) previously has supported only 2 service routines:
  - LOAD
  - DELETE

- As of z/OS R11 more service routines will be supported.
  - GETSTORE
  - FREESTORE
  - MSGRTN

- All these service routines are analogous to those routines in AMODE 31 Preinitialization (CEEPIPI).
HEAPPOOLS Diagnostic Enhancements

Enhancements to HEAPPOOLS (and HEAPPOOLS 64) diagnostics

- Format the heap pools structures and storage using IPCS
- Format the heap pools trace with finer granularity
- Limit the heap pools trace to specific pools
- Control the size of the heap pools trace
HEAPPOOLS Diagnostic Enhancements

Enhancements to HEAPPOOLS (and HEAPPOOLS 64) diagnostics

- Changes to the HEAPCHK run-time option
  - Four (4) new sub-options are added to the HEAPCHK run-time option
    - Default values provide the same behavior as in prior releases
  - These sub-options control:
    - The number of trace entries per pool (size of the trace)
    - The pool(s) to be traced
HEAPPOOLS Diagnostic Enhancements

Enhancements to HEAPPOOLS (and HEAPPOOLS 64) diagnostics

- Syntax

```plaintext
| OFF |
>>-HEAPChk--(---------,-------------------------------------->
| ON |

>---------,--------,----------,----------,------------------->
| frequency- | delay- | call depth- | pool call depth- |

>---------,--------,----------,----------,------------------->
| num of entries- | pool number- |

>---------,--------,----------,----------,------------------->
| num of entries 31- | pool number 31- |
```
HEAPPOOLS Diagnostic Enhancements

Enhancements to HEAPPOOLS (and HEAPPOOLS 64) diagnostics

- **Number of Entries**
  - Specifies the number of entries to be recorded in the heap pool trace table for the main user heap in the application. If the heap pool trace table is available and Number of Entries is 0, then the heap pool trace table is not generated.

- **Pool Number**
  - Filter the entries of heap pool trace table recording only those entries of a specific poolid for the main user heap in the application. The value should be a valid pool number (1-12). If heap pool trace table is available and Pool Number is 0 then, the entries of all pools will be traced.
HEAPPOOLS Diagnostic Enhancements

Enhancements to HEAPPOOLS (and HEAPPOOLS 64) diagnostics

- **IPCS – Formatting the heap pools trace**
  - `HPT(value) | HPTTCB(value) | HPTCELL(value) | HPTLOC(value)`
    - **HPT** - (existing keyword)
      - If the value is 0 or *, the trace for every heappools poolid is formatted. If the value is a single number (1-12), the trace for the specific heappools poolid is formatted.
    - **HPTTCB**
      - Filters the heappool trace table (if available) printing only those entries for a given TCB address (value).
    - **HPTCELL**
      - Filters the heappool trace table (if available) printing only those entries for a given cell address (value).
HEAPPOOLS Diagnostic Enhancements

Enhancements to HEAPPOOLS (and HEAPPOOLS 64) diagnostics

- **IPCS – Formatting the heap pools trace**
  - `HPT(value) | HPTTTCB (value) | HPTCELL(value) | HPTLOC(value)`
  - **HPTLOC**
    - Filters the heappool trace table (if available) printing only those entries for a given virtual storage location (value). The valid values are the following:
      - 31: Display entries located on virtual storage below the bar
      - 64: Display entries located on virtual storage above the bar
      - ALL: Entries located on virtual storage below / above the bar
    - **NOTE:** Filter options without specifying HPT implies HPT(*).
HEAPPOOLS Diagnostic Enhancements

- IPCS heap pools report
  - Formatted when HEAP or ALL is specified
  - The Heappool report will be very similar to the Heap Report.
  - The report will contain the following information:
    - QPCB
    - QPCB Entry for each pool
    - Addresses
    - Free chain validation
    - Extent validation:
      - Address and size of extent
      - Each free and allocated cell
Sources for Additional Information

- Language Environment Debugging Guide
- Language Environment Run-Time Messages
- Language Environment Programming Reference
- Language Environment Programming Guide
- Language Environment Programming Guide for 64-bit Virtual Addressing Mode
- Language Environment Customization
- Language Environment Run-Time Application Migration Guide
- Language Environment Writing ILC Applications
- Language Environment Vendor Interfaces
- Language Environment Concepts Guide
- MVS IPCS Commands
- CICS Supplied Transactions