

# Language Environment for z/VSE

– Pieces of News, Tips and Enhancements –

( LVC – Part 1 )

Wolfgang Bosch, IBM Labor Böblingen, email: [wbosch@de.ibm.com](mailto:wbosch@de.ibm.com)



<http://www.ibm.com/zVSE>

<http://twitter.com/IBMzVSE>

Thursday, October 24th, 2013



**The following are trademarks of the International Business Machines Corporation in the United States, other countries, or both.**

Not all common law marks used by IBM are listed on this page. Failure of a mark to appear does not mean that IBM does not use the mark nor does it mean that the product is not actively marketed or is not significant within its relevant market.

Those trademarks followed by ® are registered trademarks of IBM in the United States; all others are trademarks or common law marks of IBM in the United States.

For a complete list of IBM Trademarks, see [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml):

\*, AS/400®, e business (logo)®, DBE, ESCO, eServer, FICON, IBM®, IBM (logo)®, iSeries®, MVS, OS/390®, pSeries®, RS/6000®, S/30, VM/ESA®, VSE/ESA, WebSphere®, xSeries®, z/OS®, zSeries®, z/VM®, System i, System i5, System p, System p5, System x, System z, System z9®, BladeCenter®

**The following are trademarks or registered trademarks of other companies.**

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency, which is now part of the Office of Government Commerce.

\* All other products may be trademarks or registered trademarks of their respective companies.

**Notes:**

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

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

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

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

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

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

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



## Notice Regarding Specialty Engines (e.g., zIIPs, zAAPs and IFLs):

Any information contained in this document regarding Specialty Engines ("SEs") and SE eligible workloads provides only general descriptions of the types and portions of workloads that are eligible for execution on Specialty Engines (e.g., zIIPs, zAAPs, and IFLs). IBM authorizes customers to use IBM SE only to execute the processing of Eligible Workloads of specific Programs expressly authorized by IBM as specified in the "Authorized Use Table for IBM Machines" provided at [www.ibm.com/systems/support/machine\\_warranties/machine\\_code/aut.html](http://www.ibm.com/systems/support/machine_warranties/machine_code/aut.html) ("AUT").

No other workload processing is authorized for execution on an SE.

IBM offers SEs at a lower price than General Processors/Central Processors because customers are authorized to use SEs only to process certain types and/or amounts of workloads as specified by IBM in the AUT.



## Agenda ( LVC - Part 1 )



- **Module 1.1: LE/VSE Bird view, Recap and more ... ( p.5 ff. )**
- **Module 1.2: LE/VSE Enhancements with z/VSE 5.1 ( p.14 ff. )**
- **Module 1.3: Tools Reference ( p.25 ff. )**



## Module 1.1



- **LE/VSE Bird View, Recap and more ...**

- Product Infrastructure, Capabilities ( p.6 )
- Language Support ( p.7 )
- Callable Services ( p.8, 9 )
- Run-Time Options ( p.10 )
- Assembler and HLL Programming / snapshots ( p.11 )
- Debugging Start Points + Approaches / snapshots ( p.12,13 )

- **Abbreviations (used herein):**

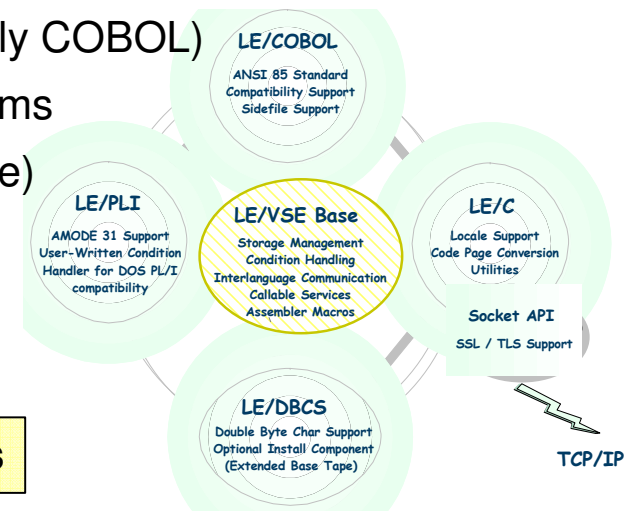
- AF: VSE/Advanced Functions (basic system control, including the supervisor and system programs)
- AR: Attention Routine (control instance to read/process console commands, resp. initiate a system service request)
- CEL: Common Execution Library (set of basic run-time services to support execution of language programs)
- HLL: High Level Language (programming language above assembler and below program generator/query languages)
- ILC: InterLanguage Communication (for applications involving routines written in different programming languages)
- LE: Language Environment (run-time environment for HLL applications)
- OLPD: Online Problem Determination (incident reporting provided via the VSE Interactive Interface(IUI) component)



## LE for z/VSE - Product Infrastructure and Capabilities

### • Some pieces that make out Language Environment for z/VSE ...

- Common Execution Library (CEL), complemented by ...
- Language-specific run-time libraries (LE/C, LE/COBOL, LE/PLI)
- Shared Callable Service Interface
- Common Functions (storage, condition, message handling ...)
- Numerous customization options (routine/language-specific, environmental)
- Coincident calling conventions and rules for implementing data structures, error handling and interfacing with system services, library routines and subsystems
- Choices for more efficient and flexible programming (+ simplify language coexistence)
- Compatibility for legacy applications (defined conditions/mainly COBOL)
- Interface for debugging High Level Languages (HLLs) programs
- Functions beyond former run-times (e.g. device independence)
- Various utilities (e.g. code page conversions, locales ...)
- Interfaces to various other products (AF, CICS, DB2, DLI, DT/VSE, EGL programs, SORT, TCP/IP, VSAM)



**In sum: the point of intersection for z/VSE Applications**



## Languages and HLASM Support

- **LE for z/VSE is needed to run applications built with the following compilers ...**

- IBM C for VSE/ESA (C/VSE)
- IBM COBOL for VSE/ESA (COBOL/VSE)
- IBM PLI for VSE/VSE (PLI/VSE)



- **High Level Assembler (HLASM) routines can also run with LE for z/VSE**

**\* solely or in combination with HLL routines \***

- Presuming they have been prepared for communication with the run-time ...
- Make use of certain conventions (safeguard for LE/VSE conformity)
- Identify themselves in the application context, e.g. by indicating "I am": MAIN=YES|NO
- Call other HLL routines in a standard manner
- This is best followed by use of product supplied assembler macros like:
  - ★ CEEENTRY/TERM (prolog/epilog),
  - ★ CEECAA|CIB|DSA|PPA (generate infrastructure mappings),
  - ★ CEEFETCH/CEERELES (dynamic load/release conforming routines),
  - ★ CEEGLOB (product level information)





## Callable Services



- Programmers can choose from a variety of available services ...

Area	Some Representatives
Condition Handling:	CEE5CIB (pointer info block); CEEGPID (version/platform id)
Date and Time:	CEEDATE (in: lilian, out: char date); CEEDYWK (day of week ← lilian)
Dynamic Storage:	CEEGTST (allocate heap w/ID); CEECRHP (allocate additional heap);
General:	CEE5DMP (take dump for diagnostics); CEEDLYM (delay processing)
Initialization/Termination:	CEE5ABD (terminate with abend); CEE5GRC/SRC (get/set enclave RC)
Locales:	CEEFTDS (format time/date->char string); CEEFMON (format monetary strings)
Math:	CEESxABS (absolute value); CEESxDIM (positive difference between numbers);
Message Handling:	CEEMGET (retrieve, format + store msg); CEEMOUT (dispatch user defined msg)
National Language:	CEE5CTY (change/query country setting)

### CEEGPID

Retrieves the version ID and the platform ID of the version and platform of LE/VSE currently in use.

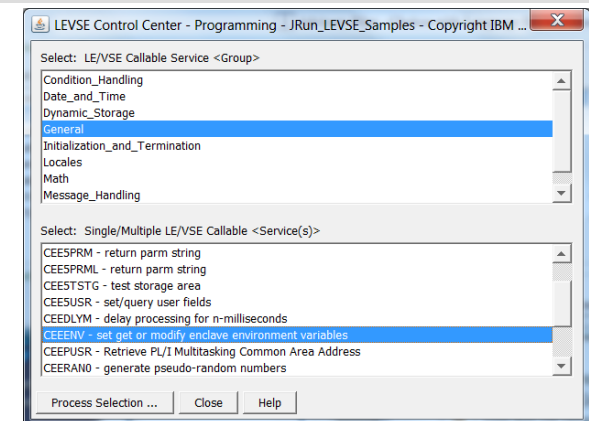
```
>> _CEEGPID_ ( _CEE_Version_ID_ , _Plat_ID_ , _fc_ ) _____ <<
```



### CEE5DMP

Generates a dump of the run-time environment of LE/VSE and the member language libraries.

```
>> _CEE5DMP_ ( _title_ , _options_ , _fc_ ) _____ >
```







```

*PROCESS MACRO;
/*****
/* CICS MODULE: CPLIDYWK (derived from shipped/BATCH: IBMDYWK.P) */
/*****
/* Function: CEEDYWK calc day of week on which Lilian date falls. **/
/*****
PLIDYWK: PROC OPTIONS(MAIN);

  %INCLUDE CEEIBMAY;
  %INCLUDE CEEIBMCT;
  DCL LILIAN INT4 ;
  DCL DAYNUM INT4 ;
  DCL 01 FC FEEDBACK ;
  DCL TMP_STR1 CHAR(72); /* temp string/output */
  DCL TMP_STR2 CHAR(72); /* temp string/output */
  LILIAN = 152385; /* input as Lilian date */

  PUT SKIP LIST ( 'PLDW/CIBMDYWK: PLI/CICS application started' );
  PUT SKIP LIST ( 'PLDW/CIBMDYWK: input Lilian date: ' || LILIAN );

  /* Call CEEDYWK, calculate day of week for Lilian input */
  PUT SKIP LIST ( 'PLDW/CIBMDYWK: call CEEDYWK service to '
    || 'calculate corresponding day of the week' );
  CALL CEEDYWK ( LILIAN , DAYNUM , FC );

  IF FBCECHK( FC, CEE000) THEN /* CEEDYWK call successful */
  DO;
    PUT SKIP LIST ( 'PLDW/CPLIDYWK: Lilian date: ' || LILIAN
      || ' ... falls on a ' );
    SELECT (DAYNUM);
      WHEN (1) PUT LIST ( 'Sunday.' );
      WHEN (2) PUT LIST ( 'Monday.' );
      WHEN (3) PUT LIST ( 'Tuesday.' );
      WHEN (4) PUT LIST ( 'Wednesday.' );
      WHEN (5) PUT LIST ( 'Thursday.' );
      WHEN (6) PUT LIST ( 'Friday.' );
      WHEN (7) PUT LIST ( 'Saturday.' );
    END /* Case of DAYNUM */;
    PUT SKIP LIST('PLCK/CPLIDYWK: *** TEST SUCCESSFUL ***');
  END;
  ELSE /* CEEDYWK call unsuccessful */
  DO;
    PUT SKIP LIST ( 'PLDW/CPLIDYWK: CEEDYWK failed with msg '
      || FC.MsgNo );
    PUT SKIP LIST( 'PLDW/CPLIDYWK: *** TEST FAILED ***' );
    /* Write 1:1 to console */
    TMP_STR1 = 'PLDW/CPLIDYWK: CEEDYWK failed with msg
      || FC.MsgNo;
    TMP_STR2 = 'PLDW/CPLIDYWK: *** TEST FAILED ***';
    EXEC CICS WRITE OPERATOR TEXT(TMP_STR1);
    EXEC CICS WRITE OPERATOR TEXT(TMP_STR2);
  END;

  /* Common at end */
  EXEC CICS SEND CONTROL FREEKB;
  EXEC CICS RETURN;
END PLIDYWK;

```

## Callable Services

Example:  
**CEEDYWK - Calculate Day of Week from Lilian**  
 The Lilian date is the number of days since 14 October 1582.



### \* CICS SYSLST log/output \*

```

A000PLDW 20130918114938 PLDW/CIBMDYWK: PLI/CICS application started
A000PLDW 20130918114938 PLDW/CIBMDYWK: input Lilian date: 152385
A000PLDW 20130918114938 PLDW/CIBMDYWK: call CEEDYWK service to calculate corresponding day of the week
A000PLDW 20130918114938 PLDW/CPLIDYWK: Lilian date: 152385 ... falls on a Saturday.

```

### \* Please also note / consider \*



- PLIDYWK sample is running as a CICS transaction (derived from LE/VSE supplied batch type sample program: IBMDYWK.P)
- You may wish to consider further, dependent programming enhancements like: each “Friday” call application x, each “Monday” start VSE Connector Server etc.
- Similar actions imaginable on a timely basis by combining “triggers” (like feedback from CEEDYWK service) with exploitation of VSE POWER based time event scheduling ...
- Subject and related functions are implicitly provided with LE/VSE. You do NOT need to duplicate them by writing an own routine !



## Run-Time Options



Area	Major Representatives
Conditional	<b>ABPerc</b> (exempt event); <b>DEPthcondlmt</b> (nesting level); <b>ERrcount</b> (# conditions tolerated); <b>TRAP</b> (enable/disable exception handling); <b>USrhdlr</b> (register use handler); <b>XUFLOW</b> (exp.overflow);
Diagnostic	<b>ABTERmenc</b> (termination type); <b>MSGFile</b> (destination); <b>TERmthdact</b> (dump level); <b>TRACE</b> (library);
Environmental	<b>ENVAR</b> (env vars, access: getenv() or CEEENV); <b>TEST</b> (give Debug Tool control)
National	<b>COUNTRY</b> (format for date,time,symbols,separators); <b>NATlang</b> (nat.language);
Program Storage	<b>HEAP</b> (allocate, how managed); <b>HEAPCHK</b> (verify)
Reports	<b>RPTOpts</b> (subject options in place); <b>RPTStg</b> (application storage in use)
Storage Mgmt	<b>ALL31</b> (addr./run mode); <b>ANYheap/Belowheap</b> (LE/VSE library heap); <b>LIBSTACK</b> (thread/save areas); <b>STACK</b> (auto vars, temp work areas);
C only	<b>ARGPARSE</b> (command line); <b>ENV</b> (op.env.); <b>EXECOPS</b> (cmd line Y N); <b>REDIR</b> (stdin/err/out);
COBOL only	<b>AIXBLD</b> (complete file/index); <b>CHeck</b> (index/subscript/ref.range); <b>CBLOPTS</b> (format arg string); <b>CBLPshpop</b> (CICS PUSH/POP HANDLE subroutine); <b>DEBUG</b> (batch/USE FOR DEBUGGING); <b>RETZero</b> (user RC 0); <b>RTEREUS</b> (reusable env. for 1st); <b>UPSI</b> (switches for COBOL routines);
PLI only	no dedicated, however please be aware of those with influence outlined below !

### • Some beneficial settings (language dependent)... better verify the actuals !

- COBOL users:            **CBLOPTS (ON)**; **CHECK(ON)**; **STORAGE(00,NONE,NONE,32K)**
- PLI users:                **DEPTHCONDLMT(0)**; **ERRCOUNT(0)**; **STORAGE(00,NONE,CLEAR,32K)**

↙ Compatibility with DOS PL/I Optimizing Compiler

Please also see: [ftp://public.dhe.ibm.com/eserver/zseries/zos/vse/pdf3/LE\\_Useful\\_Run\\_Time\\_Options.pdf](ftp://public.dhe.ibm.com/eserver/zseries/zos/vse/pdf3/LE_Useful_Run_Time_Options.pdf)



## Programming – Overview Macro Use / Program Calls / ILC



Area	Macro / Language Elements	Frame Conditions / HLL call options ...
Assembler main **	CEEENTRY(MAIN=YES),CEETERM	for use in any LE environment; CEEFETCH macro
Assembler sub *	CEEENTRY(MAIN=NO),CEETERM	for use in any LE environment; CEEFETCH macro
Assembler sub	EDCPRLG/EPIL macros	for execution in LE/C environment (bridging)
Assembler (non-LE/VSE conform)	CEEPIPI (preinitialization)	Use CDLOAD macro to load CEEPIPI table, define table entries via CEEXPITY macro for dyn. HLL load !
COBOL/VSE	CALL	Static or dynamic call options
PLI/VSE	FETCH followed by CALL *	Dynamic call
C/VSE	fetch() *	Prefer dynamic call (writable static)
TCPIP Socket *	EZASMI macro EZASOKET TCP callable functions	for HLASM programs / OS/390 source compatible for HLASM, COBOL/VSE, PLI/VSE / OS/390 compat.
CICS	EXEC CICS LINK   XCTL (preferable if switching language)	Compile units must use LE/VSE Conforming Compilers

**Note: prefer CEEFETCH macro over CEELOAD (deprecated) !**

\* Batch + CICS environment

\*\* Prefer simple HLASM main coding under CICS (due to limited CICS condition handling support for MAIN=YES) ... and better consider calling an HLL subroutine for performing the „major“ work !

[ftp://public.dhe.ibm.com/eserver/zseries/zos/vse/pdf3/LE\\_Assembler\\_main\\_under\\_CICS.pdf](ftp://public.dhe.ibm.com/eserver/zseries/zos/vse/pdf3/LE_Assembler_main_under_CICS.pdf)

Samples reference: Coding Techniques for Mixed Language Applications under LE/VSE (incl.Assembler)



## LE/VSE Debugging – Start Points ...



Evidence	Registered Data	CONFIG	Remarks / More Details
Area messages	CEE,CEL,EDC,IBM,IGZ -prefixed	TER(MSG) *	Language dependent console or SYSLST messages
Abend codes	U1xxx, U4xxx	n/a	Complementary part in LE/VSE messages
Traceback	Event call history log (read from bottom to top)	TER(TRACE) *	Event call history (user & LE/VSE modules): Stack frames, program and entry point address/offset, statement #, call status
Dump	Control block (CB), storage/file perspective	TER(DUMP) * TER(UADUMP) *	Storage of run-time event (on enclave basis): Parameter, registers, variables of active routines, file & condition info, process CBs
RC's (in CICS environment)	1xxxx, 3xxxx, 5xxxx, 10xxxx	n/a	if LE/VSE unable to generate msg, component in charge might pass back return code to CICS (usually presented by CICS on console)
C utility msgs and RC's	EDC-prefixed	n/a	For prelinker-, localedef-, iconv-, genxlt-, uconddef- and DSECT utility
z/VSE Console Reports	Info: Default Run-Time Options, Exit and Status	n/a	<b>Attention Routine (AR) commands:</b> D CEE,CEEEXIT CEESTAT CEECOPT CEEDOPT
Compile/link list of application	Complete view language element coding + build	Language specific options at compile	Often suites to verify/isolate run-time errors or „phaenomes“ ...

- Messages and Codes: <http://publibz.boulder.ibm.com/cgi-bin/bookmgr/BOOKS/FL2DRE0A/3.0>
- Dump and Traceback: <http://publibz.boulder.ibm.com/cgi-bin/bookmgr/BOOKS/FL2DRE0A/1.3.3>



## LE/VSE Debugging Approaches - Overview



Debug Focus	Symptom Type	Frame Conditions	Further Remarks/Options
Common Anchor Area	Commonly useful	All resources anchored here	Reg12 points to CAA
Data Values	Data exceptions		
Stack Frame	U4083	Problem with stack linkage, all compiled procs/blocks chained, Use of Assembler routines ?	Reg13 addressing most recently active stack; inspect "Machine State"
Storage Condition	U4088, U4093	Storage or initialization issue	Enough GETVIS ?
Termination Condition	U4094	Storage/stack ok? Pot. overlay? Use language debug features e.g. CBL SSRANGE (COBOL)	CEETRACE, Debug Tool
Condition Information Block	CEE0374C	LE/VSE condition handler creating CIB for each event	
Generate dump		Call CEE5DMP from application	
Exempt event		Set ABPERC run-time option	
Inspect CICS trace	*EXC*	CICS transaction dump with internal (or auxtrace) available	IUI providing additional OLPD records for analysis

### Some more that may help:

<http://publibz.boulder.ibm.com/cgi-bin/bookmgr/BOOKS/FL2DRE0A/1.3.3>

- MSG CEE3321C outlines the VSE perspective (with event cancel and interruption codes)
- MSG CEE3322C refers U4xxx abend codes with general type of error experienced
- MSG CEE3250C registers details about probable CICS or user abend codes
- U4xxx abend codes are complemented by reason codes with further details
- LE/VSE may raise CANCEL SVC (X'0A32') for severe problems (pointed to by PSW) !
- Collect all LE/VSE messages, traceback and dump information available (tailor for if required) !



## Module 1.2



- **LE/VSE Functional Enhancements with z/VSE 5.1**

- CEE5INF Callable Service Sample for COBOL/VSE ( p.15)
- PL/I Multitasking Run-time Options and Callable Service CEEPUSR ( p.16-17 )
- System Programmer C Environment ( p.18 )
- Enhanced LE/C Support for VSAM SHR(4) Files ( p.19 )
- TCP Callable Functions, Socket API and IPv6 ( p.20 )
- Miscellaneous Enhancements ( p.21 )
- Documentation References ( p.22 )
- Hints and Tips for z/VSE 5.1 Migration ( p.23 )
- APARs for z/VSE 5.1 ( p.24 )



## CEE5INF Callable Service Sample for COBOL



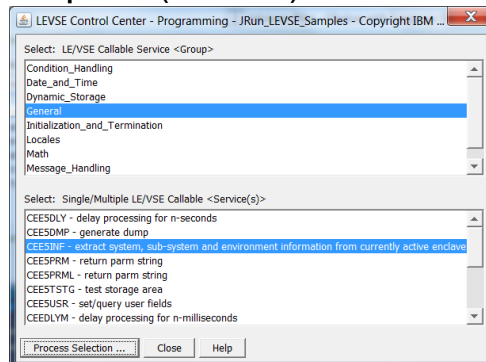
- **Callable Service:** « **CEE5INF** » (new sample for COBOL language !)  
 – Extract System, Sub-System and Environment Info From Currently-Active Enclave

**Syntax**

```

▶▶ CEE5INF (—sys_sub—, —env_inf—, —
▶ mem_id—, —gpid—, —fbc—) ▶▶
  
```

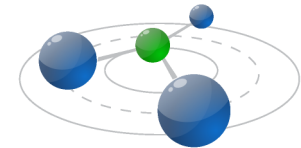
- Shipped sample **IGZT5INF.C** available in PRD2.SCEEBASE library
- In a wider scope allows the writing of « **single source** » COBOL/VSE programs for target “batch” and “CICS” environments ! (details will follow in subject LVC, Part 2)
- Also see: z/Journal article from 12/2011:  
<http://enterprisesystemsmedia.com/article/common-cobol-vse-applications-for-use-in-cics-and-batch-environments>
- The new sample is accessible via « **LEVSE\_Control\_Center V3.0** », too
  - Please see: <http://www-03.ibm.com/systems/z/os/zvse/downloads/tools.html#lecc>
  - Invoke: JRun\_LEVSE\_Samples (button) -> General -> CEE5INF (to give it a try)







## PL/I Multitasking – Complementary Callable Service –



- **New Callable Service:** « **CEEPUSR** » (PL/I only)

- Allows to retrieve the address of a pre-allocated storage area (common task area)

Syntax

```

▶▶ CEEPUSR(—function_code—, —————▶
▶—area_pointer—, —area_length—, —fc—)▶▶
  
```

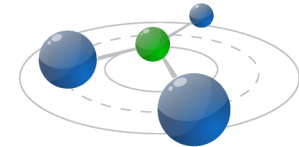
- Common task area used by PL/I multitasking applications for inter-task communication
- It's size is determined by JCL SETPARM variable « **TASKCOM** »
  - Example: // SETPARM TASKCOM=22K => up to 3 digits with K or M suffix !
  - In above context LE/VSE will attempt to allocate the requested storage size for the user
- After allocation this area is under full responsibility of programmer (not LE !)
- Shipped sample **IBMPUSR.P** available in PRD2.SCEEBASE library
- This service is not available or supported under CICS !
- Please remember PTF for **APAR PM17894** (PL/I VSE/ESA compiler) is prereq. for use !
- The new sample is accessible via « **LEVSE\_Control\_Center V3.0** », too
  - Please see: <http://www-03.ibm.com/systems/z/os/zvse/downloads/tools.html#lecc>
  - Invoke: JRun\_LEVSE\_Samples -> Dynamic Storage -> CEEPUSR (to give it a try)







## PL/I Multitasking – Run-Time Option Considerations –



### • How LE/VSE Options Affect Processing Behaviour:

- Run-time option or storage reports (generated in a fetched sub-task) will report « **IBMESTUB** » as enclave name, instead invoking sub-task program name

```
// EXEC IBMIVPMT,SIZE=512K,PARM='RPTSTG(ON)/JCL' PARM INFORMATION'
1S54I PHASE IBMIVPMT IS TO BE FETCHED FROM PRD2.WBOSCH
Fetch CEE5PRML returned parms are : "JCL PARM INFORMATION"
Options Report for Enclave IBMESTUB 07/28/11 8:53:20 AM
Language Environment for z/VSE V1 R4.8
```

LAST WHERE SET	OPTION
Installation default	ABPERC(NONE)
Programmer default	ABTERMENC(ABEND)
Installation default	NOAIXBLD
Installation default	ALL31(OFF)
Installation default	ANYHEAP(16384,8192,ANYWHERE,FREE)
Installation default	BELOWHEAP(8192,4096,FREE)
Installation default	CBLOPTS(ON)
Installation default	CBLPSHPOP(OFF)
Programmer default	CHECK(ON)

- **Fetchable PL/I programs** executing in a multi-tasking environment use the LE/VSE system-wide default BATCH run-time options !
- **To change behaviour ...**
  - Override the default options by creating a CEEUOPT.OBJ member, later included at linkedit time
  - This CEEUOPT will only be used if the fetchable routine is executed within a multi-tasking environment
  - Fetchable PL/I program tasks inherit JCL specified run-time option overrides





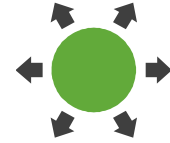
## System Programming C (SPC) Environment



- **Key message:**
  - Subject use is intended and recommended for maintenance of existing applications !
- **Significant updates in « LE/VSE V1R4.8 C Run-Time Programming Guide, SC33-6688 »**
  - Section: **Using Functions** in the SPC Environment
    - sprintf() function: no support of floating point conversion specifiers (e,E,f,g or G)
  - Section: **Creating Freestanding Applications**
    - Cannot do arithmetic's using long double or float variables
  - Section: **Initializing a Freestanding Application**
    - exit() and sprintf() functions
    - Table 36: "Parts Used for Freestanding Applications" was complemented
  - Other: Missing **JCL** and **INCLUDE information** was added
- **Linkage Recommendations**
  - To eliminate inclusion of CEESTART, use the "NOSTART" C/VSE compiler option
  - EDC0XSPC include book (forces CEESTART via EDCXCEE => might be unintended!)
- **Updated SPC samples** (JCL and source books)
  - EDCJN018.Z, EDCJN019.Z
  - EDCJL089.Z, EDCJL090.Z
  - EDXJL097.A
  - EDCJL098.A, EDCJL099.A



## Enhanced LE/C Support for VSAM SHR(4) Files



### • General Considerations

- **LE/VSE C Run-Time** does not ...
  - Provide support for read-integrity and write-integrity !
- **Data integrity on concurrent VSAM reads and writes** (use of common buffers)
  - Ensured by specifying the » **dsn= keyword** » on all calls to fopen() function
- **How about supported VSAM SHR(4) files opened in update mode ?**
  - It's programmer's responsibility to handle any potential control-interval concurrent-access locks !

### • VSAM KSDS/RRDS SHR(4) files are supported:

- For LE/C app's to exploit VSAM SHR(4), if opened in sep. partitions / diff. access modes
- if LE/C run-time "seek" or "positioning" functions are not used
- or if option "noseek" is used when the file is opened (fopen / freopen functions)

### • VSAM ESDS SHR(4) files are supported:



- Only RBA-based access can be used
- Also see LE/VSE V1R4.8 C Run-Time Programming Guide, SC33-6688





## TCP/IP Callable Functions



- The **LE/VSE C Run-Time Socket API** was enhanced to support IPv6.
  - New functions introduced with IPv6 support are:
    - **inet\_ntop, inet\_pton** (convert internet address format binary ↔ text)
    - **getaddrinfo, getnameinfo** (socket address ↔ node name and service location)
    - **freeaddrinfo** (free addrinfo structure returned by getaddrinfo)
    - **gai\_strerror** (text string describing the error)
    - if\_freenameindex, if\_indextoname, if\_nameindex, if\_nametoindex (network interface mapping)
  - Note:
    - No TCP/IP or SSL related routine is functionally integrated into the C Run-Time library. The function call is routed to be served by the TCP/IP implementation.
- Manual **LE/VSE V1R4.8 C Run-Time Library Reference, SC33-6689**
  - Provides an **overview** of the callable TCP/IP and SSL functions. 
- Manual **z/VSE TCP/IP Support, SC34-2640** includes:
  - Detailed functional **description** for each TCP/IP and SSL Callable Function. 
    - Please refer to: Chapter: TCP/IP Support for the LE/VSE C Socket Interface.
  - Also details on Linux Fast Path and IPv6/VSE.



## Miscellaneous Enhancements



- **LE/C Applications and Creation of VSE Librarian Members (with DATA=YES parameter)**
  - “DATA=YES” will be supported as an **optional** fopen() parameter
  - Any other specification on “data=” keyword will result in the fopen() failure
  - For backward compatibility “data=no” behavior will remain the default when processing librarian members !
  - Omitting optional “data=yes” parameter will invoke the default DATA=NO behavior
- **Using \_\_last\_op Codes**
  - LE/VSE V1R4.8 C Run-Time Programming Guide, SC33-6688
  - Chapter 14, Table 27 ... now includes the **latest** \_\_last\_op codes
  - It has also been updated to include the *decimal values* of the \_\_last\_op code
- **New Messages ...**
  - **CEE3216S - CEE3219S** => related to IEEE exceptions (ports from LE for z/OS)
  - **CEE3228S, CEE3229S** => related to IEEE exceptions (ports from LE for z/OS)
  - **CEE3902W** => related to new CEEPUSR callable service !
- **Replaced Message**
  - **CEE3220S** => related to IEEE exceptions (ported from LE for z/OS)





## z/VSE 5.1 Documentation – Summary Reference –



- **Bird's eye view of new LE/VSE 1.4.8 functions**
  - z/VSE V5R1.0 Release Guide, SC33-8300
- **CEEPUSR – Callable Service to Pre-Allocate Task Storage Area (PL/1 Multitasking)**
  - LE/VSE V1R4.8 Programming Reference, SC33-6685
- **Use of Fetchable Programs (PL/1 Multitasking Environment)**
  - LE/VSE V1R4.8 Programming Guide, SC33-6684
- **Major Updates to the « System Programming C » (SPC) Environment**
  - LE/VSE V1R4.8 C Run-Time Programming Guide, SC33-6688
- **Configuring « LE/C TCPIP Socket API Multiplexer » for IPv6/VSE (+ overview table)**
  - LE/VSE V1R4.8 C Run-Time Library Reference, SC33-6689
- **Function fopen() to Create VSE Librarian Members (with DATA=YES parameter)**
  - LE/VSE V1R4.8 C Run-Time Library Reference, SC33-6689
- **New Messages: CEE3216S-CEE3219S, CEE3228S, CEE3229S, CEE3902W, CEE3220S (replaced)**
  - LE/VSE V1R4.8 Debugging and Run-Time Messages Guide, SC33-6681

• In general: Languages + z/VSE

▪ **Also see: z/VSE Basics, SC24-7436**

-> Chapter 8, 9 and Appendix D

<http://www.redbooks.ibm.com/abstracts/sq247436.html>



## Hints and Tips for z/VSE 5.1 Migration

- **Instantly ensure service for PM67737 being present** ... before relinking any COBOL program !
- **LE/VSE Attention Routine Interface + Commands** ...
  - LE/VSE is shipped pre-customized and activated !
  - Please **keep it enabled**, is prereq for LE/VSE option, exit & status reports !
  - System USERBG.PROC must be current and contain the following statement:
 

```
// PWR PRELEASE RDR,CEEWARC LE - AR INTERFACE
```
- **Run-Time Option Customization**
  - Use current JCL to (re-)apply run-time option changes ...
    - Skeletons: CEEWDOPT + CEEWCOPT (ICCF 62)
  - Backlevel LE/VSE option modules in place will be indicated by z/VSE system !
    - **U4092 RSN42** (batch) or **RC11060** abend (at CICS-init time)
- **Supplied LE/CICS Transaction**
  - Don't use « CLER » for storage run-time option changes « in-flight » (scope = all transid's !)
  - For changing other run-opts « CLER » may suit as preferred agent of choice (quick change)
- **Supplied SVA Loadlists** (\$SVACEE; \$SVAEDC/M; \$SVAIBM/M; \$SVAIGZ/M)
  - Apply those load list(s) best matching your application / environment needs !
- **CICS Sub-System**
  - BMS users with map corruptions to think of LE/VSE run-opt: STORAGE=(00,00,CLEAR,0K)
  - CICS SIT value RUWAPPOOL=YES (to maximize storage for LE/VSE applications)
    - Particularly useful in case of heavy EXEC CICS LINK exploitation



## Recommended LE/VSE 1.4.8 APARs (Reference)



### • LE/VSE 1.4.8 APAR history (recommended for installation)

– PM88622	LE/Base	ABENDU4087 RC02 OCCURS DURING EXEC CICS RETURN	
– PM86172	LE/C	FLOCATE() RETURNS RC=0 INSTEAD RC=-1 / VSAM KSDS non-existing rec	
– PM86062	LE/Base	APPLICATION-SPEC. RUN-TIME OPTION OVERRIDES NOT HONORED	E512
– PM81989	LE/Base	PL/I GOTO TO INTERNAL LABEL CAUSES ABEND UNDER BATCH	
– PM77159	LECOB	EXTRN SYMPTOM FOR IGZESTUB IN LNKEDT MAP	
– PM73461	LE/Base	ABEND0C4 IN CEECREIN FOLLOWING AN AEXY ABEND	
– PM74318	LE/PLI	MSG IBM0122I 'ONCODE'=22 'RECORD' CONDITION / VSAM ESDS	
– PM73473	LE/Base	MORE COMPREHENSIVE DSA CHECKING ON PL/1	
– PM72051	LE/COB	MODULE ILBDCMM0 GOT EXTRN SYMPTOM IN LNKEDT MAP	
– PM67737	LE/COB	CORR. R2 AFTER COBOL CALL / UNPREDICTABLE APP.FAILURES	
– PM53860	LE/Base	MSG CEE3322C AB4087 IN BATCH OR CICS / CEE2503S / UTC/GMT	E511
– PM52953	LE/COB	FILE STATUS 35 EXPERIENCED ON VSAM ESDS FILE(REUSE)	
– PM51783	LE/PLI	IBM0482I ONCODE=310 FIXEDOVERFLOW CONDITION	
– PM51170	LE/C	FLOCATE(), SHR(4) Keyed-access files inadvertently disabled (errno: 12)	







## Module 1.3

### ▪ **Recommended Tools Levels (for use with z/VSE 5.1)**

- CEETRACE - Version 1.2.0b ( p.26 )
- LEVSE\_Control\_Center - Version 3.0 ( p. 27 )





## Latest Tool Level – CEETRACE ( V1.2.0b)

<http://www-03.ibm.com/systems/z/os/zvse/downloads/tools.html#ceetrace>

### • New Trace Capabilities introduced for z/VSE 5.1

#### – Auto Report feature ...

- Allows specification of <entry point name> and <statement number> at which auto-generation of execution history report takes place ... or ...
- Repeated execution history report generation after specified number of statements

#### – High Level Language (HLL) Statement Exit

- Allow exit module to be called for each HLL statement (CEETRACE enabled app)
- Sample CELHLLXT.Z supplied via LE z/VSE installation library (demonstrates count of executed HLL statements, issuing simple console message)

#### – Mini Dump Option

- For unhandled conditions, a small formatted dump with Condition Information Block (CIB) and Machine State Block (MCH) info can be included in execution statement history report

### • New Related Configuration Options

#### – Via CEETRACE.INI file (AR command override possible) ...

- AUTO\_RPRT=Rnnnn | Snnnn | OFF
- EXIT\_MOD=entry point name | OFF
- MINI\_DUMP=YES | NO

### • Includes on top fixes for z/VSE 5.1

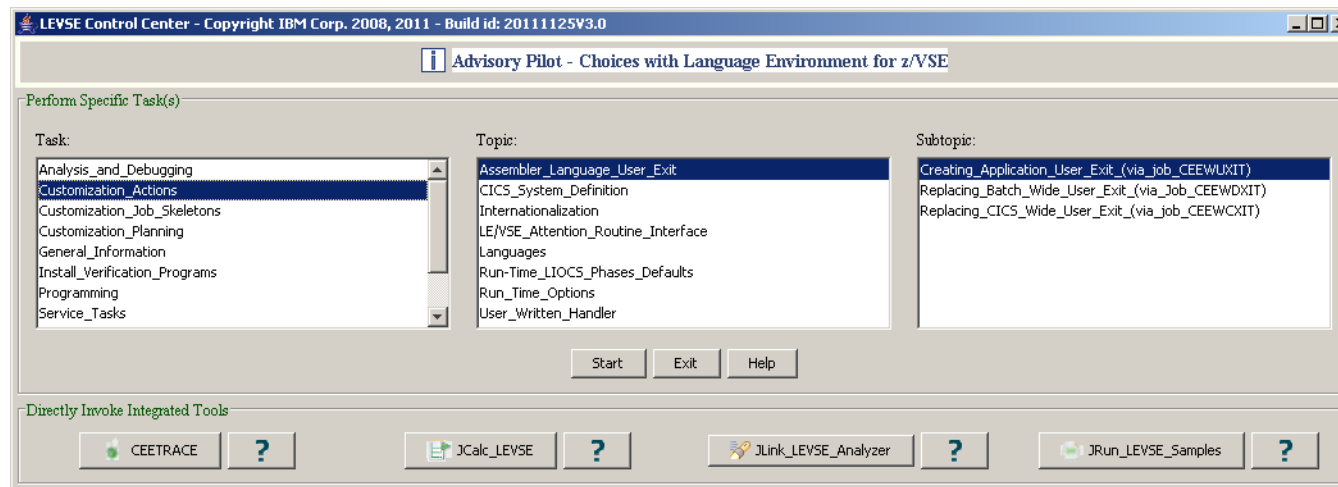
- See download file: « CEETRACE\_V1R2.0b-READ-ME.txt » ... for details ...



## Latest Tool Level – LEVSE Control Center (V3.0)

<http://www-03.ibm.com/systems/z/os/zvse/downloads/tools.html#lecc>

- « **GUI front-end** » to explore **CEETRACE** more easily
  - Call button on main window
- « **Continued Integration** » of **LE/VSE Enhancements (mainly z/VSE 5.1)**
  - CEETRACE feature (also see: page #27) ...
    - Auto-Report Feature, HLL Statement Exit, Mini Dump capabilities
  - Added Attention Routine Overrides for above
    - S CEE,CEETRACE= ...
  - Callable Services
    - Existent CEE5INF service: new COBOL sample IGZT5INF.C
    - New CEEPUSR service (PL/I only): sample IBMPUSR.P
    - CEE5MC2 (from z/VSE 4.3): new sample EDC5MC2.C



# Language Environment for z/VSE - New Features, Tips and Abend Analysis.

( LVC – Part 2 )

Mr Garry Hasler – IBM Australia, Perth, Aust. Development Lab.  
Email : [ghasler@au1.ibm.com](mailto:ghasler@au1.ibm.com)



<http://www.ibm.com/zVSE>  
<http://twitter.com/IBMzVSE>

Thursday, October 24th, 2013



**The following are trademarks of the International Business Machines Corporation in the United States, other countries, or both.**

Not all common law marks used by IBM are listed on this page. Failure of a mark to appear does not mean that IBM does not use the mark nor does it mean that the product is not actively marketed or is not significant within its relevant market.

Those trademarks followed by ® are registered trademarks of IBM in the United States; all others are trademarks or common law marks of IBM in the United States.

For a complete list of IBM Trademarks, see [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml):

\*, AS/400®, e business(logo)®, DBE, ESCO, eServer, FICON, IBM®, IBM (logo)®, iSeries®, MVS, OS/390®, pSeries®, RS/6000®, S/30, VM/ESA®, VSE/ESA, WebSphere®, xSeries®, z/OS®, zSeries®, z/VM®, System i, System i5, System p, System p5, System x, System z, System z9®, BladeCenter®

**The following are trademarks or registered trademarks of other companies.**

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency, which is now part of the Office of Government Commerce.

\* All other products may be trademarks or registered trademarks of their respective companies.

**Notes:**

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

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

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

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

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

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

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



## Notice Regarding Specialty Engines (e.g., zIIPs, zAAPs and IFLs):

Any information contained in this document regarding Specialty Engines ("SEs") and SE eligible workloads provides only general descriptions of the types and portions of workloads that are eligible for execution on Specialty Engines (e.g., zIIPs, zAAPs, and IFLs). IBM authorizes customers to use IBM SE only to execute the processing of Eligible Workloads of specific Programs expressly authorized by IBM as specified in the "Authorized Use Table for IBM Machines" provided at [www.ibm.com/systems/support/machine\\_warranties/machine\\_code/aut.html](http://www.ibm.com/systems/support/machine_warranties/machine_code/aut.html) ("AUT").

No other workload processing is authorized for execution on an SE.

IBM offers SEs at a lower price than General Processors/Central Processors because customers are authorized to use SEs only to process certain types and/or amounts of workloads as specified by IBM in the AUT.

# Agenda



- **Environment Independent Application Execution.**
  - How to use LE z/VSE callable services in a COBOL/VSE program to provide environment independent application execution.
    - Eg Execution of the same COBOL routine in both CICS and BATCH environments.
- **Understanding 4083 User Abends.**
  - Why do they occur?
  - What are they?
  - How to handle them.
    - Using the 4083 Trace-Back Report.
- **CEETRACE feature Updates.**
  - Environment validation – HEAP storage corruption analysis.
  - Using the new “auto report” function.
- **Questions?**



# **Environment Independent Execution**





## Environment Independent Application Execution.

- Some current techniques available with LE z/VSE to determine an applications current execution environment.
  - CEE5INF – Extract System information from the active enclave.
    - Introduced with LE/VSE 1.4.5 (z/VSE 4.1)
    - Provides sub-system, LE enclave, language and LE z/VSE product information.
    - Information is provided as bit-flags within a full-word for each section.
    - Multiple language examples are provided in the LE z/VSE Programming Reference for how to use.
  - iscics() – LE/C run-time library function. Returns “true” if executing under CICS or “false” if not running under CICS. Available only for C/VSE applications use.
  - LE-enabled assembler routines can use the CEECAASSCIC value of the CEECAASBSYS field in the active R12/CEECAA (see CEECAA.A macro).
- So how do we construct a simple COBOL/VSE routine that can execute independent of its environment?



## Environment Independent Application Execution (contd).

Using CEE5INF :

- COBOL example :
  - Parameter definitions
  - Call the service to extract environment information
  - Interrogate returned parameters.
  - Direct application execution flow based upon currently active execution environment.

```
01 bit-values      PIC S9(9) BINARY.
01 bit-result     PIC S9(9) BINARY.
01 sys-inf        PIC S9(9) BINARY.      System environment info.
01 env-inf        PIC S9(9) BINARY.      Language Environment info.
01 mem-id         PIC S9(9) BINARY.      Language(s) information.
01 gpid           PIC S9(9) BINARY.      LE production information.
01 Environ       PIC X.
                88 CICS                VALUE X'01'.
                88 BATCH                VALUE X'02'.
.....
Call 'CEE5INF' Using sys-inf, env-inf, mem-id, gpid, fc.
```

## Environment Independent Application Execution (contd).

Using CEE5INF (contd) :

- Next problem - CEE5INF returns field contents as bit strings.
  - Can be easier to handle with assembler, PL/1 or C/VSE, bit more tricky with COBOL/VSE.
- LE z/VSE callable services provide this ability to COBOL easily.
  - For our situation – CEESITST – test bit value.
  - “sys-inf” is the bit-string we want to test
  - “bit-value” is bit number of “sys-inf” to test.
    - Bit 31 = CICS indication flag.
  - “fc” is a standard feedback code parameter.
  - “bit-result” is the test result.

Combine this and the previous code section into a paragraph for processing during program initialization

```
Move 31 to bit-value.  
Call 'CEESITST' Using sys-inf, bit-value, fc, bit-result.  
If bit-result = 1 then  
    Set CICS to True  
Else  
    Set BATCH to True  
End-if.
```



## Environment Independent Application Execution (contd).

- Using CEE5INF (contd) :
  - Now we can use the “CICS” or “BATCH” flag to control our programs execution.
    - For example :

```
    If BATCH then
      DISPLAY ' --> Running in BATCH Environment. ' UPON CONSOLE.
      Perform 0200-BATCH
      Goback
    End-if.

    If CICS then
      EXEC CICS WRITE OPERATOR TEXT(ENV-MSG)
           TEXTLENGTH(MSG-LEN) END-EXEC
      Perform 0300-CICS
      EXEC CICS RETURN END-EXEC
    End-if.
```



## Environment Independent Application Execution Summary.

### Points to remember:

- Always determine execution environment (using CEE5INF) first.
- Split application into two paths. CICS and BATCH.
- Ensure both pathways (ie entire program) adhere to the CICS environment limitations.
  - Eg restricted verbs (see CICS/TS Application Programmers Guide) etc.
    - Consider enabling and using the WORD(CICS) COBOL/VSE compiler option.
      - » See section 1.2.4.3.2 - CICS Reserved Word Table (IGYCCICS) - in the COBOL/VSE and VisualAge COBOL MLE for VSE Installation and Customization Guide (SC26-8071-01).
    - Abend with message IGZ011C may be issued in CICS environment otherwise.
- Compile the COBOL routine with RENT and DATA(31) options.
- Translate the program using the CICS translator with XOPTS(COBOL3).
- Ensure your link-edit step includes the CICS language independent interface stub – DFHELII.
  - Do not over-ride the compiler and link-editor determined entry-point or AMODE/RMODE!
- A complete sample COBOL program code and build JCL is available from the z/VSE Web site
  - <ftp://public.dhe.ibm.com/eserver/zseries/zos/vse/download/COBJOURN.txt>
- The article associated with using the above sample can be found here:
  - <http://enterprisesystemsmidia.com/article/common-cobol-vse-applications-for-use-in-cics-and-batch-environments>

**Important Note :** The CICS translator will add CICS parameters to your PROCEDURE DIVISION.  
May require application review for handling of any passed parameters

# Understanding 4083 User Abends

## Understanding 4083 User Abends.

### ▪ Why do 4083 Abends Occur?

- Language Environment structure is dependent upon correctly chained DSAs.
- Without correct DSA chaining application execution issues can occur :
  - Handling of non-severe (informational or warning) conditions by languages compromised.
  - Unable to produce a comprehensive “Trace-Back” Report in CEE5DMP output.
  - Integrity of stack storage (automatic storage) is now compromised (NAB).
  - Any earlier registered condition handlers unable to be called.
  - Unable to identify active members (languages) or routines on the current stack etc, etc...
- What drives the 4083 abend?
  - Initially something (an unhandled condition) must cause Language Environment to verify the current DSA chain.
  - This can be, but not limited to :
    - Any unhandled condition (eg program-check, I/O error etc).
    - A dump request is made that requires a trace-back report.
    - A language function (verb or built-in function) requires active stack members information.
  - When the currently active DSA chain is examined, if any issues are found, a 4083 abend is issued with the corresponding reason code depending upon the error found in the DSA chain.
- **Applicable only to BATCH. Is not issued under CICS.**



## Understanding 4083 User Abends.

### ▪ What are “4083” ( Back-Chain In Error ) Abends?

- First we need to have a basic understanding of LE z/VSE Stack Storage.
- What is User Stack Storage? :
  - Created automatically upon entry to LE-enabled applications (eg prolog code).
  - Driven by LE conforming compilers and LE provided assembler macros.
  - Dependent upon CAA address-ability in R12 and an initialized LE z/VSE environment.
  - Contains Dynamic Save Areas (DSA). Also known as “Stack Frames”.
  - Conforms to the LE z/VSE DSA layout.
    - Includes standard S/390 linkage register save area.
    - See member CEEDSA.A in your LE z/VSE Installation Sub-library.
  - Contains automatic storage (local variables) for the application. For example :
    - Eg C/VSE : int i;      PL/I : DCL i fixed;
    - **Is NOT used for COBOL/VSE working-storage!**
      - » **But! The LE/COBOL run-time and LE itself rely upon stack storage.**
  - Initial allocation size, location and initial value are controlled via STACK/ALL31 and STORAGE run-time options.

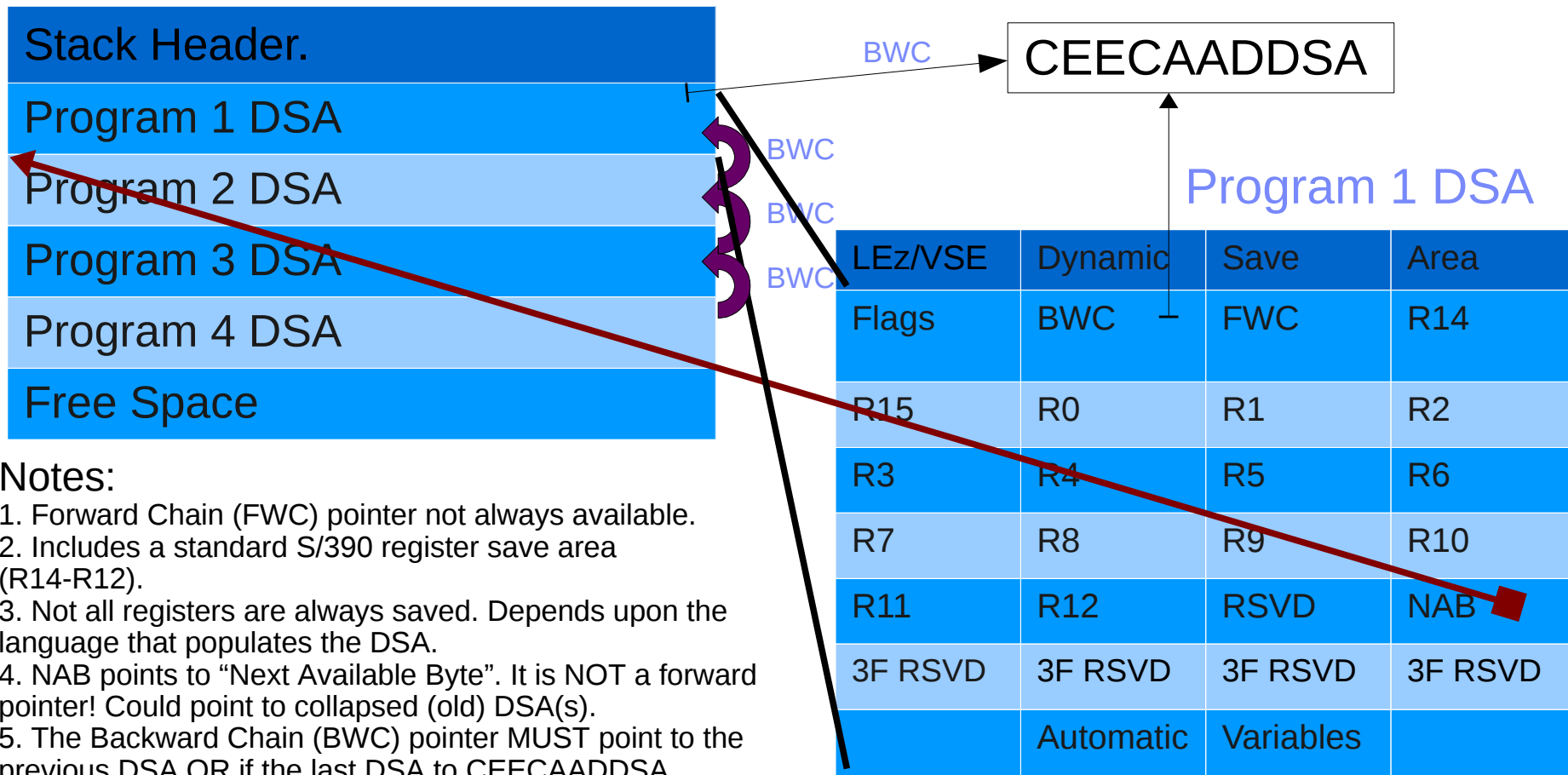




## Understanding 4083 User Abends.

### What are “4083” ( Back-Chain In Error ) Abends (contd) ?

– What does an LE z/VSE Stack Frame / DSA look like ?



#### Notes:

1. Forward Chain (FWC) pointer not always available.
2. Includes a standard S/390 register save area (R14-R12).
3. Not all registers are always saved. Depends upon the language that populates the DSA.
4. NAB points to “Next Available Byte”. It is NOT a forward pointer! Could point to collapsed (old) DSA(s).
5. The Backward Chain (BWC) pointer MUST point to the previous DSA OR if the last DSA to CEECAADDSA.

## Understanding 4083 User Abends.

### ▪ What are “4083” ( Back-Chain In Error ) Abends (contd)?

#### – Common reason codes:

- X'03' - Save area chains should end with a save area pointed to by CEECAADDSA (dummy DSA). Instead it is terminated with a zero back-chain pointer.

Sample Stack linkage in storage. Also referred to as “DSA” or “Automatic storage”.

Stor Addr	Flags	BWD	FWD	R14	R15				
005A01E8	10000000	00000000	005A0400	8058DEF6	805DBC30	00000000	005A03F0	005A01E8	.....!.Q!.!.....6.).....!.0!.!..Y
005A0208	80420144	005C3A20	005C3B58	005A0278	005C3BCA	00421840	80000000	805521D2	.....*...*...!...*.....!.....K
005A0228	804257B8	80448F78	005C0260	005A0400	00000000	00432213	10007D88	005A0018	.....*...-!.....!.....'h.!..

Flags = Reserved full word (4-bytes). Do not use – used by LE z/VSE languages internally.

BWD = Backward Pointer – Address of previous “DSA” or our callers stack storage.

FWD = Forward Pointer – Address of previously called (by us) program stack storage. Not always provided/available.

R14 = This programs caller execution return point (usually points immediately after BALR/BSM/BASSM/BASR instruction).

R15 = Target programs entry point address – where execution has been transferred to upon entry.

### Common Causes:

- Non-LE-conforming “main” assembler routine used.
- Assembler routine involved not conforming to standard S/390 linkage conventions.
- Non-LE conforming HLL language. Eg DOS/VS COBOL not link-edited with the LE z/VSE run-time.

### Diagnosis items to consider:

- R15 in invalid DSA points to entry code of possible culprit for the incorrect back-chain value.
- Storage overlay of DSA from an earlier routine could also be responsible.





## Understanding 4083 User Abends.

### ▪ What are “4083” ( Back-Chain In Error ) Abends (contd)?

#### – Common reason codes:

- X'02' - Traversal of the back chain resulted in a program check.

Sample Stack linkage in storage. Also referred to as “DSA” or “Automatic storage”.

Stor Addr	Flags	BWD	FWD	R14	R15				
005A01E8	10000000	FOFO0000	005A0400	8058DEF6	805DBC30	00000000	005A03F0	005A01E8	.....!.Q!.!.....6.).....!.0!.!..Y
005A0208	80420144	005C3A20	005C3B58	005A0278	005C3BCA	00421840	80000000	805521D2	.....*....*....!....*..... .....K
005A0228	804257B8	80448F78	005C0260	005A0400	00000000	00432213	10007D88	005A0018	.....*.-.!.....'h.!..

Flags = Reserved full word (4-bytes). Do not use – used by LE z/VSE languages internally.

BWD = Backward Pointer – Address of previous “DSA” or our callers stack storage.

FWD = Forward Pointer – Address of previously called (by us) program stack storage. Not always provided/available.

R14 = This programs caller execution return point (usually points immediately after BALR/BSM/BASSM/BASR instruction).

R15 = Target programs entry point address – where execution has been transferred to upon entry.

### Common Causes:

- Non-LE conforming or an assembler routine involved not conforming to standard S/390 linkage conventions.
- Automatic storage increased (eg copybook of an array) without increasing (ie re-compilation/assembly) DSA size.
- Non-LE conforming HLL language. Eg DOS/VS COBOL not link-edited with the LE z/VSE run-time.

### Diagnosis items to consider:

- R15 in invalid DSA points to entry code of possible culprit for the invalid back-chain address.
- Storage overlay of DSA could also be responsible.



## Understanding 4083 User Abends.

### ▪ What are “4083” ( Back-Chain In Error ) Abends (contd)?

#### – Common reason codes:

- X'01' - A save area loop exists. The save area points to itself or another save area incorrectly points to a higher save area.

Sample Stack linkage in storage. Also referred to as “DSA” or “Automatic storage”.

Stor Addr	Flags	FWD	R14	R15	
005A01E8	10000000	005A01E8	005A0400	8058DEF6	805DBC30 00000000 005A03F0 005A01E8
005A0208	80420144	005C3A20	005C3B58	005A0278	005C3BCA 00421840 80000000 805521D2
005A0228	804257B8	80448F78	005C0260	005A0400	00000000 00432213 10007D88 005A0018

.....!.Q!......6.).....!.0!.!..Y  
 .....\*....\*....!....\*..... .....K  
 .....\*.-.!.....'h.!..

Flags = Reserved full word (4-bytes). Do not use – used by LE z/VSE languages internally.

BWD = Backward Pointer – Address of previous “DSA” or our callers stack storage.

FWD = Forward Pointer – Address of previously called (by us) program stack storage. Not always provided/available.

R14 = This programs caller execution return point (usually points immediately after BALR/BSM/BASSM/BASR instruction).

R15 = Target programs entry point address – where execution has been transferred to upon entry.

**Notes:** It is not possible for LE to detect all possible forms of DSA chain loops. In situations where a loop exists but is not detected an indefinite CPU loop during condition handling processing may occur.

#### Common Causes:

- Non-LE conforming assembler or an assembler routine involved not conforming to standard S/390 linkage conventions.
- Non-reentrant dynamically loaded subroutine recursively called multiple times.

#### Diagnosis items to review:

- R15 in invalid DSA points entry code of possible culprit for the back-chain corruption.
- Check for non-reentrant dynamically introduced non-LE HLL or assembler routines that could be called multiple times.
- Verify correct save area prolog code is used.



## Understanding 4083 User Abends.

### ▪ How to Handle 4083 ( Back-Chain In Error ) Abends?

- Introduced initially with LE z/VSE 1.4.4. (z/VSE 3.1) is the “4083 Trace-back Report”.
  - A trace-back report of all consecutively accessible valid DSAs on the stack.
- Depending upon the initial cause, a further 1 or 2 formatted dumps are provided :
  - A formatted dump of the LE z/VSE Condition Information Block (CIB).
  - With z/VSE 4.3 and onwards if the original condition (prior to the 4083) is a program-check, then a formatted dump of the “Machine State” information will be produced.
- Ok, but how do I use all this information?
  - Focus on finding the original condition – not the 4083 abend itself.
  - Start with the 4083 abend trace-back report. Look for calls from your LE conforming programs (eg COBOL/VSE) to a non-LE conforming routine (shown as empty in the “Program Unit” and/or “Entry” columns).

CEE5DMP V1 R4.8: 4083 TraceBack of all Valid DSA Chains on the Active Stack

09/10/13 2:28:59 PM

Information for enclave COBVSE1

Information for thread 8000000000000000

Traceback:

DSA Addr	Program Unit	PU Addr	PU Offset	Entry	E Addr	E Offset	Statement	Service	Status
015E4420	IGZEOUT	00488700	+01148BB8	IGZEOUT	00488700	+01148BB8			Call
<b>0045E018</b>		<b>00423A00</b>	<b>+00000000</b>		<b>00423A00</b>	<b>+00000000</b>			<b>Call</b>
0047E538	<b>COBVSE1</b>	00420078	+00000AF6	<b>COBVSE1</b>	00420078	<b>+00000AF6</b>	<b>706</b>		Call

End of CEE5DMP report.



## Understanding 4083 User Abends.

- How to Handle 4083 ( Back-Chain In Error ) Abends (contd)?
  - Ok, but how do I use the other dump information provided?
    - Using the formatted dump of the LE z/VSE Condition Information Block (CIB) :

CIB for : 015E5B28

```

+000000 CIB_Eye.. CIB          CIB_Back. 00000000  CIB_Frwd. 00000000  CIB_Size. 010C      CIB_Ver.. 0008
+000010 CIB_Plat. 00000000  Reserved. 00000000  CIB_Cond. 00030C81  59C3C5C5 00000000  CIB_Mach. 015E5870
+000028 CIB_OLdc. 00000000  00000000  00000000  CIB_Flg1. 00          CIB_Flg2. 00          CIB_Flg3. 00
+000037 CIB_Flg4. 00          CIB_HDsf. 00000000  CIB_HDen. 00000000  CIB_H Drs. 00000000  CIB_RMsf. 00423B5C
+000048 CIB_RMpt. 00423B18  CIB_RSmh. 00000000  Reserved. 00000000  00000000  00000000  00000000  00000000
+000064          00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000
+000088          00000000  00000000          CIB_Vsr.. 00000000  00000000          CIB_Vsto. 00000000
+00009C CIB_Vpsa. 00000000  CIB_Mcb.. 00000000  CIB_Mrn.. 00000000  00000000          CIB_Mflg. 00
+0000AD Reserved. 0000000  CIB_flg5. 40          CIB_flg6. 00          CIB_flg7. 00          CIB_flg8. 00
+0000B4 CIB_ABcd. 00000020  CIB_ABrc. 00000001  CIB_ABnm. ....      CIB_Pl... 00000000  CIB_SV2.. 00000000
+0000CC CIB_SV1.. 00423B5C  CIB_Int.. 00423B16  CIB_Qdat. 00000000  CIB_FdBk. 00000000  CIB_Fun.. 00000000
+0000E0 CIB_Toke. 00000000  CIB_Mid.. 00000000  CIB_Stat. 00000000  CIB_Rtcc. 00000000  CIB_Ppav. 00000000
+0000F4 CIB_ABte. ....      CIB_Sdwa. 00000000
  
```

The hi-lighted CIB information tells us :

1. CIB\_Cond (Condition) = X'00030C81' : H'0003' = "Severe". H'0C81' = 3201.  
X'59C3C5C5' : X'59' = flags, X'C3C5C5' = "CEE" --> CEE3201S – Operation Exception.
2. CIB\_ABcd (Abend Code) = X'20' = Message OS03 --> A program check has occurred.  
(z/VSE Messages and Codes Vol 1 section VSE/AF Cancel Codes)
3. CIB\_SV1 (Save Area 1) = X'00423858'. This is the DSA that was found to be invalid. Will need system dump to verify.
4. CIB\_Int (Interrupt) = PSW interrupt address (X'00423B16').



## Understanding 4083 User Abends.

- How to Handle 4083 ( Back-Chain In Error ) Abends (contd)?
  - Ok, but how do I find out why the CEE3201S program-check occurred?
    - Start by using the provided Machine State dump :

Machine State: 015E5878

+000000	MCH_reg0.	0047E6A4	MCH_reg1.	0000077E	MCH_reg2.	00420B3C	MCH_reg3.	00480AB0	MCH_reg4.	00420452
+000014	MCH_reg5.	40000000	MCH_reg6.	0048041C	MCH_reg7.	00000000	MCH_reg8.	0047FD8F	MCH_reg9.	00423A00
+000028	MCH_regA.	00420190	MCH_regB.	004208B4	MCH_regC.	00440F78	MCH_regD.	00423B5C	MCH_regE.	70423A82
+00003C	MCH_regF.	00000000	<b>MCH_psw..</b>	07DD3000	<b>00423B18</b>	<b>MCH_ilc..</b>	<b>0002</b>	<b>MCH_intc.</b>	<b>0001</b>	
+00004C	MCH_Rsvd.	00000000	MCH_Fltp.	00000000	00000000	4BC5E125	0B6BE000	4E000000	000266CA	40404040
+00006C		40404040	MCH_Rsvd.	00000000	00000000	00000000	00000000	00000000	00000000	00000000
+00008C		00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
+0000B0		00000000	00000000	00000000	00000000	00000000	00000000	00000000	MCH_Bear.	00423AD2

The hi-lighted information tells us :

1. MCH\_psw indicates AMODE(24) and the instruction following the failing operation (X'00423B18')
2. MCH\_ilc (X'0002') states failing instructions length (instruction length code).
3. MCH\_intc (interrupt code) shows X'01' or Operation Exception (Equiv. of LE Message CEE3201S).

Therefore, failing instruction address = MCH\_psw – MCH\_ilc

Or : X'00423B18' – X'0002' = X'00423B16' (same as that provided in the CIB\_int field earlier).



## Understanding 4083 User Abends.

- How to Handle 4083 ( Back-Chain In Error ) Abends (contd)?
  - Ok, but how do I put all this information together?
    - Start with the 4083 Trace-Back report – review statement number of last successful call.
    - Then check compiler listing for target routine name.

```

Traceback:
 DSA Addr  Program Unit  PU Addr  PU Offset  Entry      E Addr  E Offset  Statement  Service  Status
 015E4420  IGZEOUT          00488700 +01148BB8 IGZEOUT    00488700 +01148BB8
 0045E018          00423A00 +00000000          00423A00 +00000000          Call
 0047E538  COBVSE1        00420078 +00000AF6 COBVSE1  00420078 +00000AF6      706      Call
End of CEE5DMP report.

COBOL/VSE compiler listing for COBVSE1 :
000706          CALL 'ASMPROG'.          EXT
  
```

- Next we check the LNKEDT map for the start location of ASMPROG.

```

ENTRY      AT      FACTOR  OFFSET  OFFSET
-----
COBVSE1    420078  420078  000000  000000
DOSCOB     421C80  421C80  001C08  001C08
IGZ5INF    422BC8  422BC8  002B50  002B50
ASMPROG   423A00 423A00 003988 003988
  
```

- Now using the earlier calculated failing instruction address (X'00423B16') we can calculate the failing instructions offset into ASMPROG : X'00423B16' – X'00423A00' = X'116'.





## Understanding 4083 User Abends.

### ▪ How to Handle 4083 ( Back-Chain In Error ) Abends (contd)?

- In the ASMPROG HLASM listing we look for offset X'116'.

```

Active Usings: ASMPROG,R9
Loc  Object Code      Addr1 Addr2  Stmt  Source Statement
                                00116      126 EXIT    DS    0H
000116 0000                                129    DC    H'00'
000118 17FF                                131    XR    R15,R15

Save-area corrupting code :
000000          00000 001BC   36 ASMPROG  CSECT
000000 90EC D00C          0000C   37          STM   14,12,12(13)      SAVE CALLERS REGS
000004 189F          R:9 00000   38          LR    R9,R15             GET BASE REGISTER
000006 17FF          00000   39          USING ASMPROG,R9        USE R9 AS BASE
000008 50F0 9160          00160   41          XR    R15,R15           Initialise
00000C 41F0 915C          0015C   43          ST    R15,SAVAREA+4    BACK-CHAIN Save-area
000010 50FD 0008          00008   44          LA    R15,SAVAREA       GET ROUTINES SAVE AREA ADDR
000014 18DF          00008   45          ST    R15,8(R13)       FORWARD-CHAIN SAVE AREA
                                LR    R13,R15           R13 IS ROUTINES SAVE AREA

```

### In summary :

The DC H'00' instruction at offset X'116' was executed which resulted in an Operation Exception. So we can see that the U4083 abend is a subsequent issue to the real failure. The Operation Exception was the real problem but because of a non-LE conforming assembler program corrupting the DSA chain (ASMPROG storing R15 instead of R13 in the back-chain field) LE z/VSE was unable to perform language condition handling semantics or report on the actual failure (via a formatted CEE5DMP dump with a complete Trace Back report). So was forced to issue a U4083 abend instead to terminate the enclave.

## Understanding 4083 User Abends.

- Any questions on handling 4083 User Abends?



# **CEETRACE Feature Update**





## CEETRACE Feature Update.

### Environment Validation (contd) :

- For HEAP storage CEETRACE validation processing provides :
  - Standard LE z/VSE HEAPCHK output :

```
CEE3701W Heap damage found by HEAPCHK Run-time option
CEE3710I Heap Element at 006C7020 is damaged, Expected data is: 006C7000 00400008
006C7000: C8C1D5C3 00448500 005C5000 00000000 006C7000 00000000 00400028 00000000 |HANC..e..*&.....%.....|
006C7020: F3F1F1F6 00400008 00000000 00000000 00000000 00000000 00000000 00000000 |3116.....|
```

- New Console CELT060E validation failure message.

```
Z2 0051 CELT060E ENVIRONMENT CHECKING TRAPPED CORRUPTION.
Z2 0051 CELR021W LANGUAGE ENVIRONMENT FOR Z/VSE CEETRACE REPORT COMPLETE.
```

- Program Execution Statement History Report showing corruption culprit identification

```
14/08/2013 09:15:13.56 DD:SYSIPT main 106 +000003F0 C This Language does not support the SYSDEBUG fi
14/08/2013 09:15:13.56 DD:SYSIPT main 107 +00000434 C This Language does not support the SYSDEBUG fi
14/08/2013 09:15:13.57 DD:SYSIPT main 108 +0000046E C Requested Validation failed at previous stmt#
```

- Check application compile listing at Line/Stmt #107 for error.

```
LINE STMT
107 29 | *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9.....*
      | bytes = sprintf(m_char+count, "%d\0", div_res.quot);
```

## CEETRACE Feature Update.

### ▪ Using the “auto\_rprt” CEETRACE function :

- Benefits :
  - Allows for a program execution statement history report to be produced when :
    - A specific statement number within a specific module entry-point is executed
    - Or after a specified number of statements have been executed within the entire application.
  - Provides a program execution statement history report without an abend occurring.
- Can be used to target program logic execution flow issues.
- Supported in both the BATCH and CICS environments.





## CEETRACE Feature Update.

### ▪ Using the “auto\_rprt” CEETRACE option (contd) :

– For example :

- Produce a program statement execution history report showing program logic execution flow when statement number #768 in program IGZT5INF is executed.
  - Use a current compiler listing to determine the desired statement number to monitor.

```
000767          *****
000768          Move Ver to Ver-D.
000769          Display ' LE z/VSE Version : ' Ver-D.
```

- At a z/VSE operators console issue the following commands :

– Note : specify the entry-point name – not the PHASE name. See the link-edit map for this information.

```
s cee,ceetrace=(auto_rprt_epn=igz5inf)
AR 0015 CEL4068I CEETRACE Over Ride Options Accepted
AR 0015 CEL4019I Language Environment for z/VSE command complete.
```

– Define statement number to monitor within entry point.

```
s cee,ceetrace=(auto_rprt=s768)
AR 0015 CEL4068I CEETRACE Over Ride Options Accepted
AR 0015 CEL4019I Language Environment for z/VSE command complete.
```



## CEETRACE Feature Update.

### ▪ Using the “auto\_rprt” CEETRACE option (contd) :

- The resultant CEETRACE report output when the program is executed is:

```

14/08/2013 13:57:37.90  IGZ5INF  IGZ5INF      741  +0000097C  COBOL  If not CEE000 of fc then
14/08/2013 13:57:37.90  IGZ5INF  Ent/Ext/Par  N/A.  +000009AA  COBOL  External Entry/Exit point, End clause or Par
14/08/2013 13:57:37.90  IGZ5INF  IGZ5INF      745  +000009AC  COBOL  If bit-result = 1 then
14/08/2013 13:57:37.90  IGZ5INF  Ent/Ext/Par  N/A.  +000009C2  COBOL  External Entry/Exit point, End clause or Par
14/08/2013 13:57:37.90  IGZ5INF  IGZ5INF      746  +000009C4  COBOL  Display ' Yay!!! this is COBOL/VSE '
14/08/2013 13:57:37.90  IGZ5INF  Ent/Ext/Par  N/A.  +000009F6  COBOL  External Entry/Exit point, End clause or Par
14/08/2013 13:57:37.90  IGZ5INF  IGZ5INF      753  +000009F8  COBOL  Move 18 to bit-value.
14/08/2013 13:57:37.90  IGZ5INF  IGZ5INF      754  +00000A02  COBOL  Move 'CEESITST' to LE-CWI.
14/08/2013 13:57:37.90  IGZ5INF  IGZ5INF      755  +00000A0C  COBOL  Call LE-CWI Using env-info, bit-value, fc, bit
14/08/2013 13:57:37.90  IGZ5INF  Ent/Ext/Par  N/A.  +00000A6A  COBOL  External Entry/Exit point, End clause or Par
14/08/2013 13:57:37.90  IGZ5INF  IGZ5INF      756  +00000A6C  COBOL  If not CEE000 of fc then
14/08/2013 13:57:37.90  IGZ5INF  Ent/Ext/Par  N/A.  +00000A9A  COBOL  External Entry/Exit point, End clause or Par
14/08/2013 13:57:37.90  IGZ5INF  IGZ5INF      760  +00000A9C  COBOL  If bit-result = 1 then
14/08/2013 13:57:37.90  IGZ5INF  Ent/Ext/Par  N/A.  +00000AD0  COBOL  External Entry/Exit point, End clause or Par
14/08/2013 13:57:37.90  IGZ5INF  IGZ5INF      763  +00000AD2  COBOL  Display '          running in AMODE24.'
14/08/2013 13:57:37.90  IGZ5INF  Ent/Ext/Par  N/A.  +00000AE6  COBOL  External Entry/Exit point, End clause or Par
14/08/2013 13:57:37.90  IGZ5INF  IGZ5INF      768  +00000AE8  COBOL  Move Ver to Ver-D.

```

CEETRACE Program Execution Trace Report Complete





## CEETRACE Feature Update.

### ▪ Further information

- CEETRACE Heap Storage corruption online tutorial (on vimeo) :
  - <http://vimeo.com/69933156>
- CEETRACE specific new features, hints and tips (on slideshare) :
  - <http://www.slideshare.net/lezvse/ceetrace-nnew-featurestipsandtricks>
- Latest code version and documentation for CEETRACE :
  - <http://www-03.ibm.com/systems/z/os/zvse/downloads/tools.html#ceetrace>

Спасибо

Russian

धन्यवाद

Hindi

Bedankt

Nederlands

شكراً

Arabic

Merci

French

Obrigado

Brazilian Portuguese

THANK YOU

English

Gracias!

Spanish

多谢

Simplified Chinese

Danke

German

多謝

Traditional Chinese

ありがとうございました

Japanese

감사합니다

Thank You

# Questions



Please forward your questions or remarks to :

**vsesupportLE@de.ibm.com**

LEzVSE Blog : <https://www.ibm.com/developerworks/community/blogs/lezvse/?lang=en>



## z/VSE Live Virtual Classes

ADOBE® CONNECT™

z/VSE

@ <http://www.ibm.com/zvse/education/>

LINUX + z/VM + z/VSE

@ <http://www.vm.ibm.com/education/lvc/>

Read about upcoming LVCs on @ <http://twitter.com/IBMzVSE>

Join the LVC distribution list by sending a short mail to [alina.glodowski@de.ibm.com](mailto:alina.glodowski@de.ibm.com)

