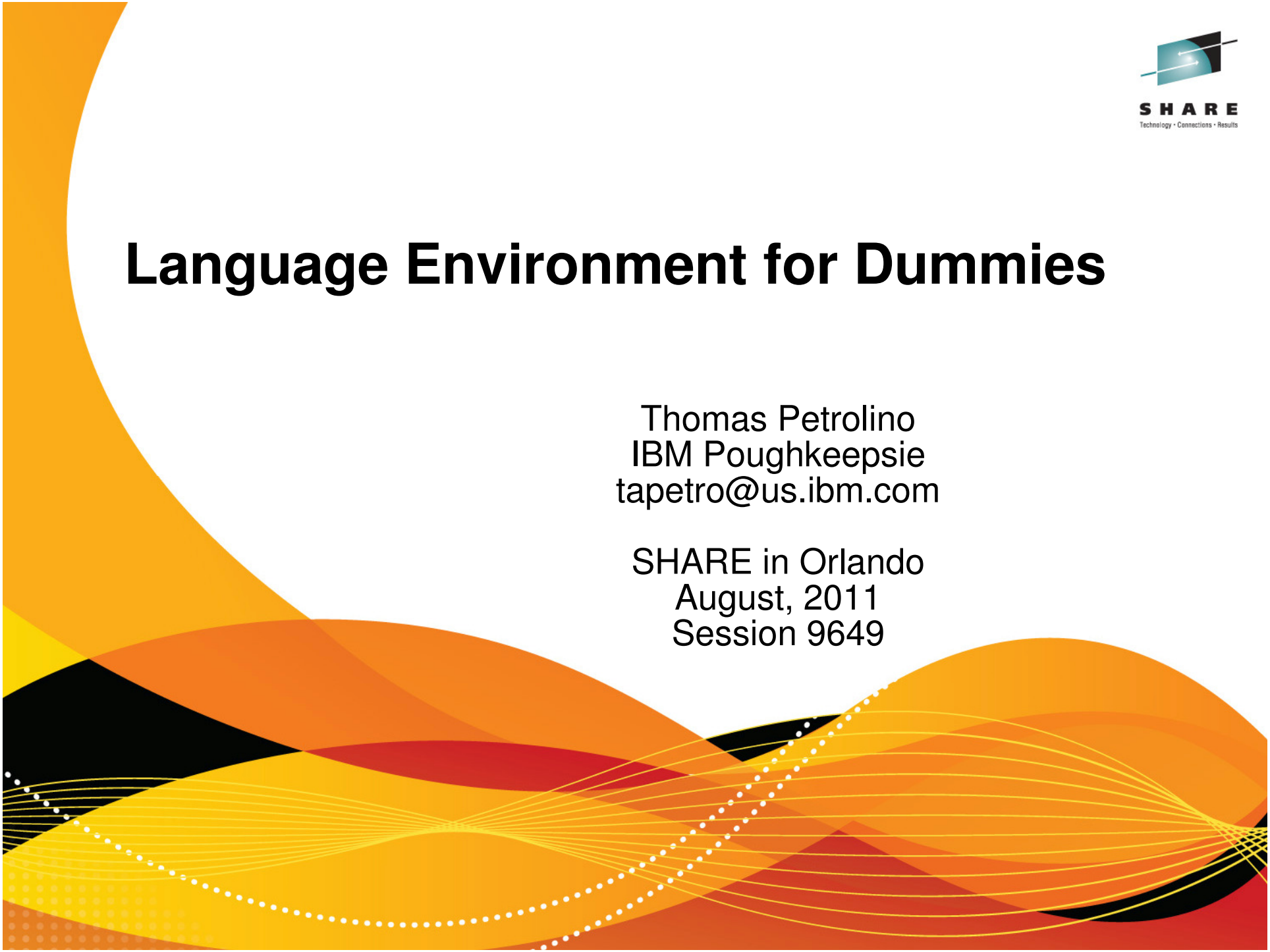


Language Environment for Dummies

Thomas Petrolino
IBM Poughkeepsie
tapetro@us.ibm.com

SHARE in Orlando
August, 2011
Session 9649

Trademarks

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

- CICS®
- DB2®
- 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.
SET and Secure Electronic Transaction are trademarks owned by SET Secure Electronic Transaction LLC.

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

Notes:

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

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

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

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

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

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

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

Agenda

- What is a Run-time Library?
- Why LE?
- LE Terminology
- LE CEL Functions
- Setting Run-time Options
- Appendix

What is a Run-Time Library?

- A Run-time Library works together with the code produced by a compiler to provide functionality for an application
  - Obtain and manage storage
  - Read and write data
  - Perform math calculations

- There are advantages to providing function in a Run-time Library
  - Greatly reduces need for the compilers to generate the code
  - Shields the languages from needing detailed knowledge of the underlying operating system and hardware
  - Greatly reduces the need to recompile and re-link when fixes are required to run-time functions

So, Why Language Environment?

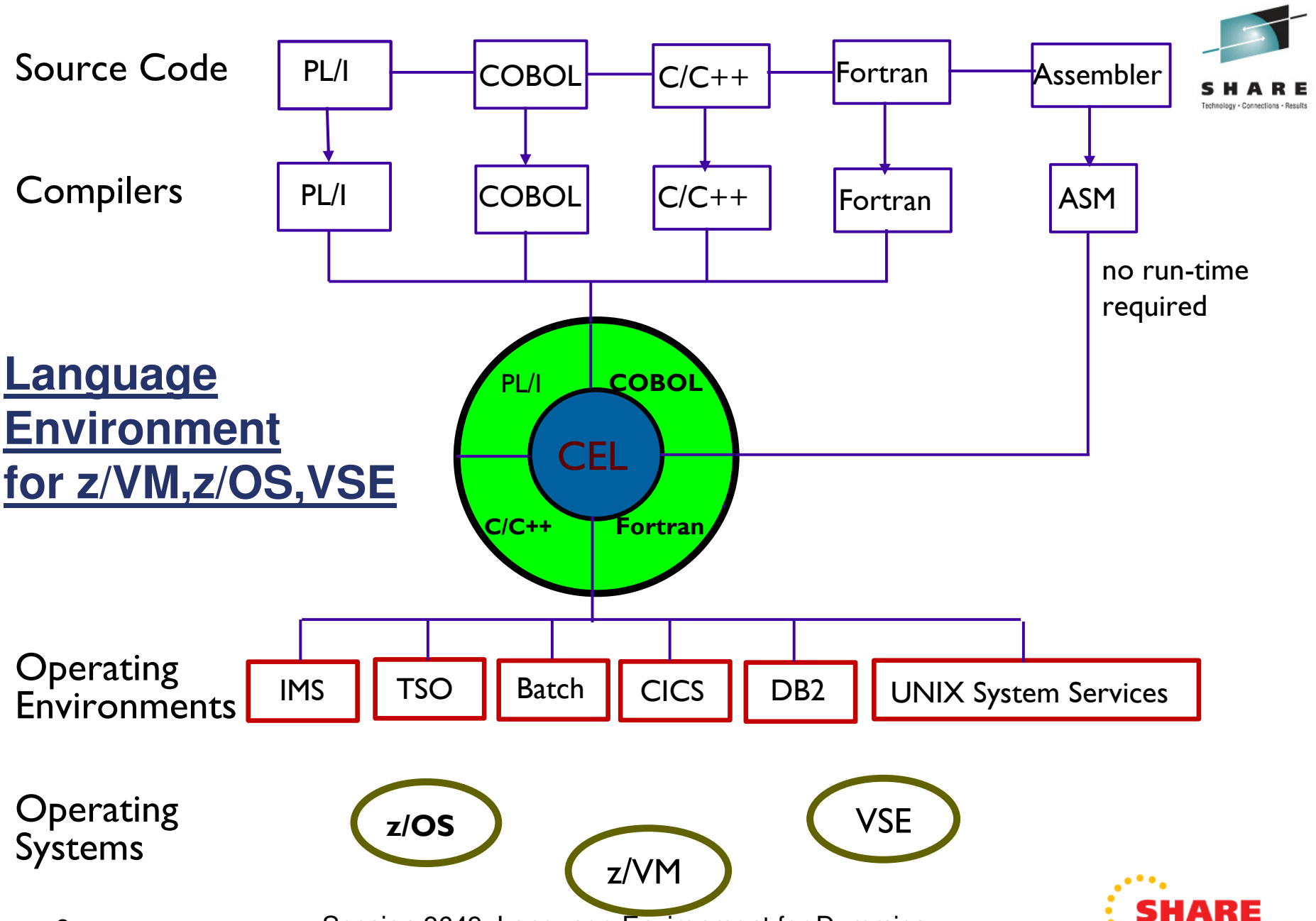
- Since their creation, customers were having trouble getting COBOL and PL/I to play nicely together
  - COBOL and PL/I each designed to be stand-alone, unaware of each other
    - When leaving a COBOL program to return to a PL/I program, the COBOL library might free storage that PL/I still wanted
    - Language-specific Math Libraries produced different results
- Customers at GUIDE and SHARE worked with IBM to design a solution
  - The result: **Language Environment**

Time to make the doughnut…

- Pre-LE environment
  - 4 independent products
  - upward incompatibilities
  - loose adherence to standards
  - purely a customer application enabler
- LE environment
  - 1 product for z/OS, z/VM and VSE
  - 100% upward/downward compatibility
  - strict adherence to standards
  - part of the z/OS base
  - exploiters include USS, TCP/IP, BCPii, LOTUS Domino, WebSphere, etc...

Other Advantages

- Language Environment not only helped the languages to cooperate with each other, but also allowed member languages to share each other's features. For example:
  - COBOL can use the C and PL/I condition handling infrastructure
  - Storage managed in a 'common' fashion
  - All languages now access the excellent Fortran library math routines
  - "hybrid" languages – Enterprise PL/I


Source Code
PL/I
COBOL
C/C++
Fortran
Assembler
Compilers
PL/I
COBOL
C/C++
Fortran
ASM
no run-time required
Language Environment for z/VM,z/OS,VSE
PL/I
COBOL
CEL
C/C++
Fortran
Operating Environments
IMS
TSO
Batch
CICS
DB2
UNIX System Services
Operating Systems
z/OS
z/VM
VSE

LE Terminology - Program Management

- **main program** – the routine that causes the LE environment to be initialized
- **routine** either a procedure, function, or subroutine
  Equivalent HLL terms:
  - COBOL - program
  - C/C++ - function
  - PL/I - procedure, BEGIN block
- **ILC** – inter-language communication – application contains a mixture of languages, which introduces special issues
  - how the languages' data maps across load module boundaries
  - how conditions are handled
  - how data can be passed and received by each language

LE Terminology - Program Management

- **member language** – a high-level language that is compiled with an LE-supported compiler
- **member event handler** - member-supplied routine that is called at various times as a program runs when a significant event has occurred, or when the environment needs some information that is held by the member
- **LE-Enabled** - Routine that can run with LE run-time, and may also run with previous run-times. Cannot make use of Language Environment callable services.
- **LE-Conforming** - Routine that can run only with the LE run-time library. Can make use of LE callable services.

LE Terminology – Callable Services

- **LE Callable Services** – programmatic way of utilizing LE services
  - AWI - Application Writer Interface
  - CWI - Compiler Writer Interface
  - CEE prefixed – general to all platforms
  - CEE3 prefixed – specific to only z/OS

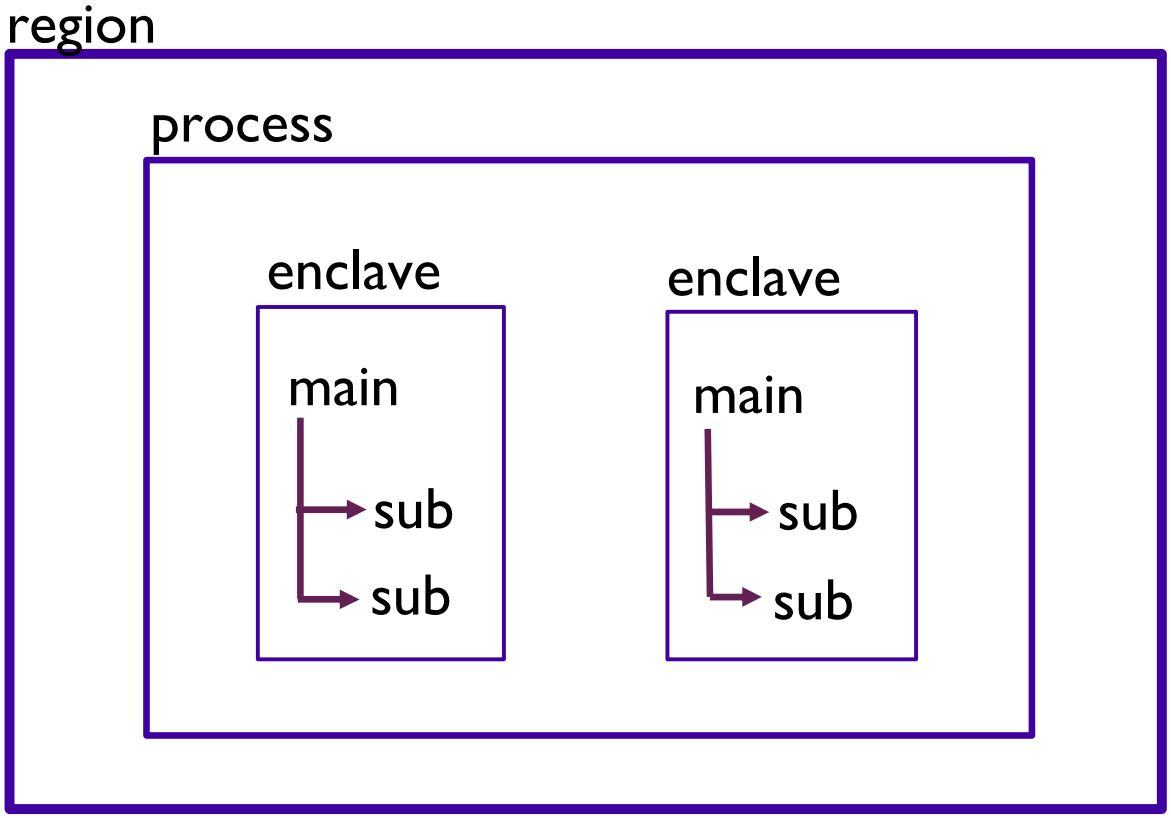
  - SHARE Session: Introducing LE Callable Services, plus a User's View of Why and How You Should Exploit Them in Your Applications – Fri 9:30AM

- **USS Assembler Callable Services –** supported by the C/C++ specific portion of the Run-time
  - BPX prefixed

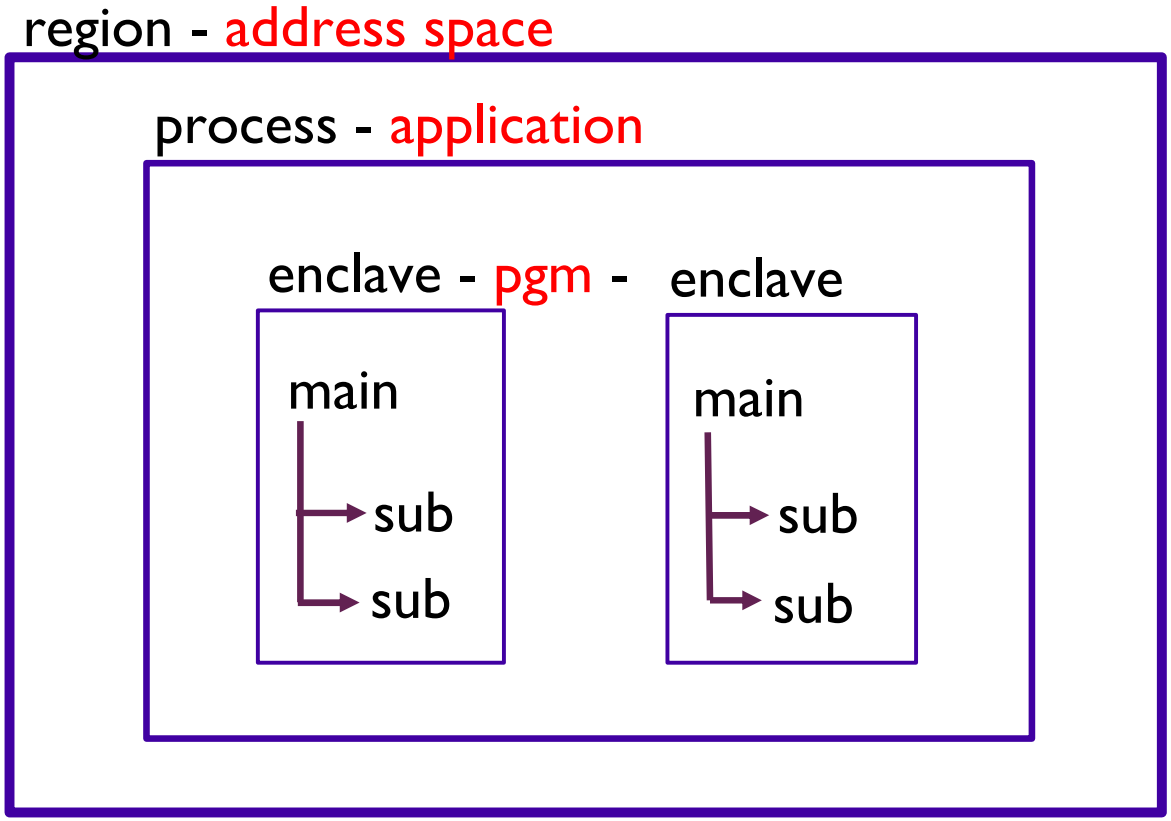
LE Terminology – Program Model

- **region** - the range of storage the application set runs in
- **process** - set of applications that accomplish a task
- **enclave** - an application - set of modules that accomplish some subtask
- **thread** - dispatchable unit of work that shares storage with others in the enclave

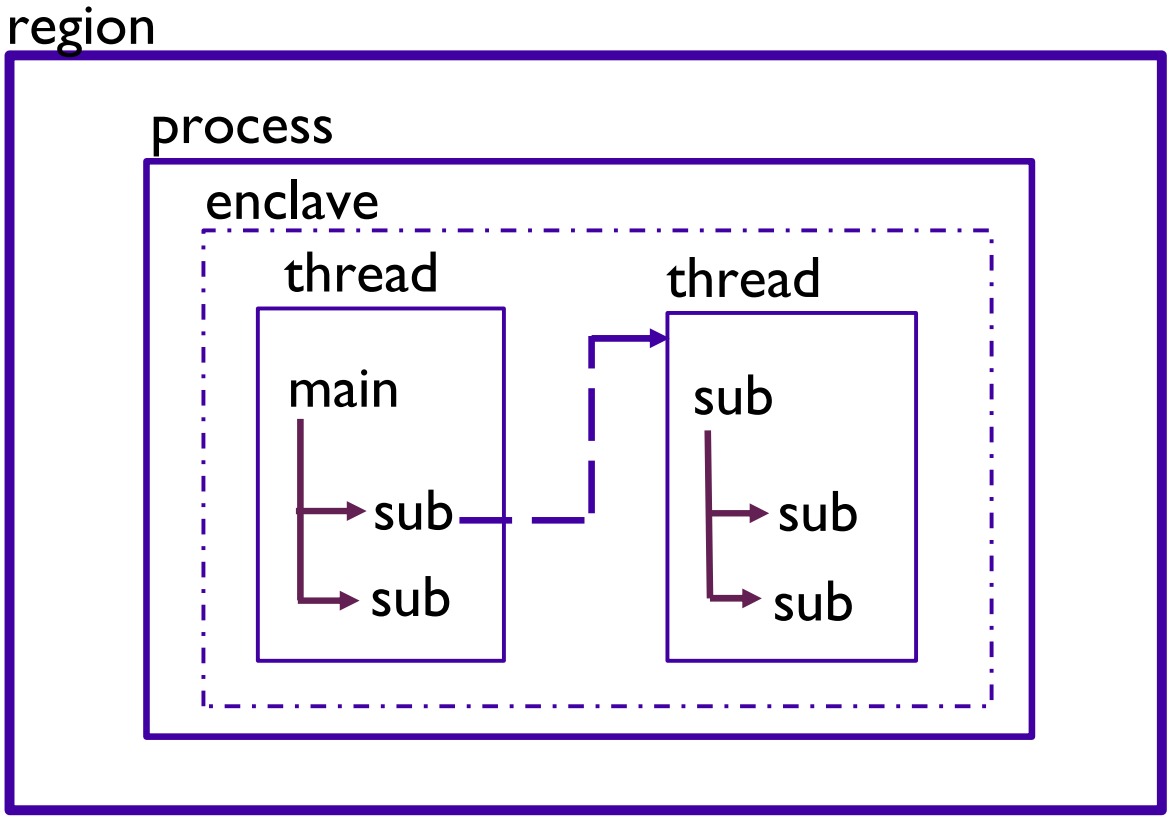
LE Terminology - Program Model



LE Terminology - MVS 'Model'



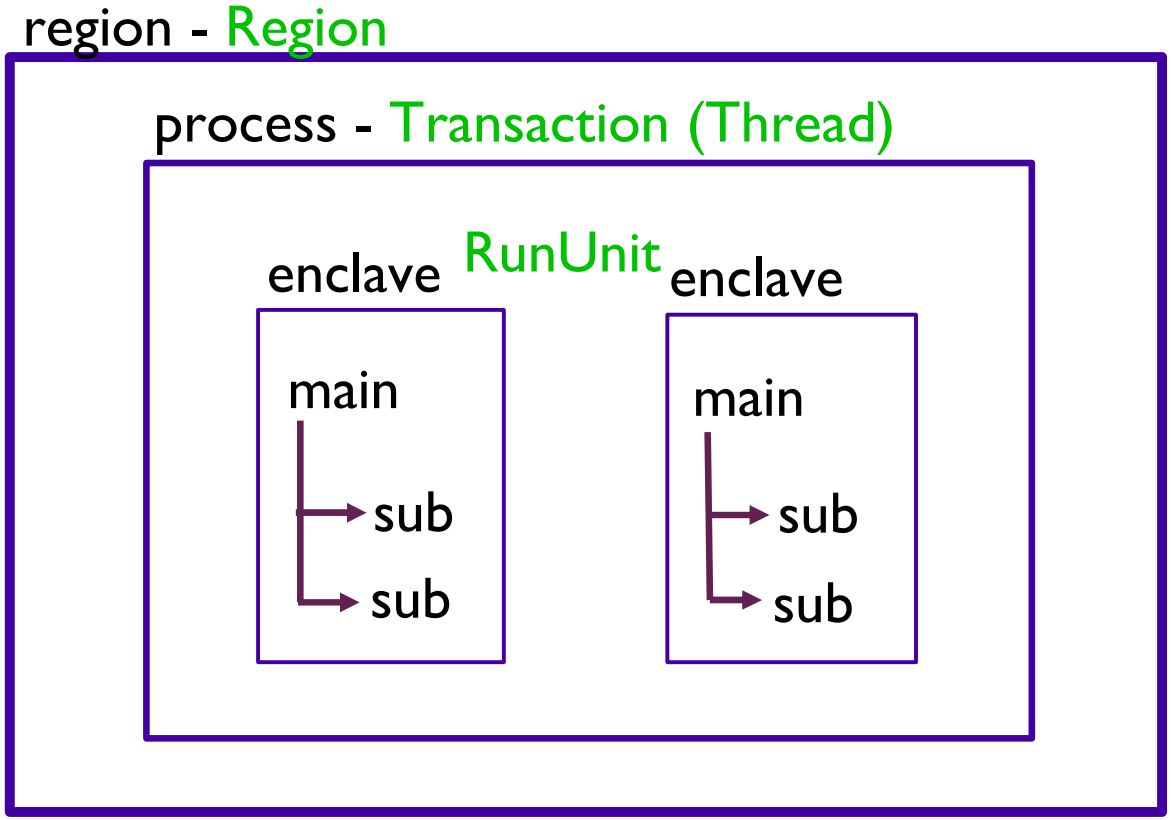
LE Terminology – Multi-threading 'Model'



CICS Terminology

- **region** - the range of storage the application set runs in
- **transaction** - set of applications that accomplish a task
- **run-unit** - an application - set of modules that accomplish some subtask

LE Terminology - CICS 'Model'



LE CEL Functions

- CEL is a set of common functions and routines used by all member languages of LE
  - Initialization/Termination
  - Storage Management
  - Condition Handling
  - Message Services
  - Date/Time Services
  - Math Functions
- Behavior customizable by the use of Run-time Options

Common LE Functions – Initialization/Termination

- LE code linked with the module begins a bootstrap process to initialize LE
  - initial storage is obtained
  - this LE instance 'registered' with UNIX System Services
  - condition handlers initialized
  - active member language specific run-time is initialized
- Control is given to the application code
- Once the application ends and 'returns' to LE
  - The LE environment is terminated
  - System resources obtained during initialization and throughout the execution of the application are cleaned up

Common LE Functions - Storage Management

- LE manages two types of storage for use by the application (and itself):
  - HEAP - used for COBOL WORKING-STORAGE, C malloc, and PL/I ALLOCATE requests
  - STACK - module linkage (save areas), C and PL/I automatic variables, COBOL LOCAL-STORAGE
- Initial storage is obtained with one GETMAIN and managed internal to LE

Common LE Functions - Condition Handling

- Condition - Any change to the normal flow of a program
  - a.k.a. exception, interruption
  - Could be detected by hardware or software (ours or yours)
- Condition Handler – A routine called by LE to respond to a condition
  - Registered by application using CEEHDLR, or part of a member language semantics, such as PL/I ON statements
- Condition Handler Response
  - Resume – after corrective action taken, control returns to a 'resume cursor'
    - Either back to point of failure, or to a new resume point set by the condition handler
  - Percolate - decline to handle the condition, LE calls next condition handler
  - Promote - change condition meaning and percolate

Common LE Functions - Condition Handling

- Diagnostic Documentation
  - messages (same as module prefixes)
    - CEE CEL
    - IGZ COBOL
    - IBM PL/I
    - AFH FORTRAN
    - EDC C/C++
  - CEEDUMP and/or system dump
  - Run-time Options Report
  - Run-time Storage Report

Common LE Functions - Condition Handling

- LE Abend Codes
  - designated as USER abends
  - U4000-4095 - reserved for applications running under LE
  - many abends codes have associated reason codes to further isolate the problem
  - some abends are the result of LE problems while others are application problems
  - ‘special’ processing needed to generate U1000 style abend codes

Common LE Functions - Message Services

- allows HLLs to 'issue' common messages
- messages written to a common place - LE's MSGFILE
- 'abstracts' system failures from the application
- can be formatted in:
  - Mixed-case American English (ENU)
  - Uppercase American English (UEN)
  - Japanese (JPN)

Common LE Functions – Date/Time Services

- provides a consistent 'answer' when requesting date and time from the running system
- format date and time by country code
- parse date and time values
- convert between different formats (Gregorian, Julian, Asian, etc)
- calculate days between dates, elapsed time
- get local time
- handle 2 year dates as part of Y2K solution

Common LE Functions – Math Services

- derived from FORTRAN math functions
- binary, single floating point, double floating point, IEEE support
- See the LE Programming Reference for a complete list

Run-Time Options

- Allows users to specify how Language Environment behaves when an application runs
  - Performance tuning
  - Error handling characteristics
  - Storage management
  - Production of debugging information
- May be set in many different locations with varying scopes

Setting Run-Time Options

- To set default RTOs for applications across all systems
  - **Installation defaults** (CEEDOPT/CEECOPT/CELQDOPT)
    - SMP/E USERMOD used to update Language Environment modules
    - Note: USERMODs will be eliminated after V1R13!
- To set default RTOs for applications on one or more systems
  - **System defaults**
    - Options specified in a PARMLIB member (CEEPRMxx)
    - Options specified with an operator command (SETCEE)
- To affect applications running within a region
  - **Region Level Overrides** (CEEROPT/CELQROPT)
    - CICS TS, LRR users (e.g. IMS), also Batch
    - Separate module loaded at run-time during region initialization
    - CLER transaction for CICS environment (RTO subset)

Setting Run-Time Options

- To provide RTO settings for a specific application:
  - **Application Level Overrides** (CEEUOPT/CELQUOPT)
    - CSECT linked with the application
  - **Programmer Overrides**
    - #pragma runopts for C/C++
    - PLIXOPT for PL/I
- To provide RTO settings for a given run of an application:
  - **Program Invocation Overrides**
    - USS shell: export _CEE_RUNOPTS='run-time options'
    - In batch, on EXEC card: PARM=
  - **DD:CEEOPTS Overrides**
    - Optional data set in which run-time options may be specified

Setting Run-Time Options

- Options Merge (priority)
  - Program Invocation Overrides
  - DD:CEEOPTS Overrides
  - Programmer Overrides
  - Application Level Overrides
  - Region Level Overrides (where applicable)
  - System Defaults (CEEPRMxx and SETCEE)
  - Installation Defaults (through V1R13)

- For more information on setting run-time options, see Appendix

Key Run-Time Options

- Subtopics
  - Tuning
    - Additional Information in SHARE sessions:
      - Look What I Found Under the Bar! (Thu 3:00PM)
  - Diagnostics
    - Additional Information in SHARE sessions:
      - Heap Damage, Is Your Insurance Up-to-Date? (Fri 8:00AM)
      - LE Crime Scene Investigation (Fri 11:00AM)

Key Run-Time Options - Tuning

- ALL31(option)
  - ON For AMODE 31 programs
  - OFF For AMODE 24 programs
    (can be determined dynamically)

Key Run-Time Options - Tuning

- ANYHEAP(initial, increment, location, disp)
- BELOWHEAP(initial, increment, disp)
- HEAP(initial, increment, location, disp, init24, incr24)
  - initial — Minimum size of initial heap segment
  - increment — Minimum size of additional segments
  - location — BELOW (<16MB), ANYWHERE
  - disp — KEEP, FREE (action when empty)
  - Notes:
    - ANYHEAP/BELOWHEAP used internally by Language Environment
    - HEAP – used for application-related storage
      - COBOL WORKING-STORAGE (for RENT programs)
      - Dynamic storage (C malloc, C++ new, PL/I ALLOCATE)

Key Run-Time Options - Tuning

- STACK(init, incr, location, disp, dsinit, dsincr)
  - init — Actual size of initial stack segment
  - incr — Minimum size of additional segments
  - location — BELOW, ANYWHERE
  - disp — KEEP, FREE (action when empty)
  - dsinit — XPLINK initial stack
  - dsincr — XPLINK increment stack
  - Notes:
    - Used for Dynamic Save Areas / Stack Frames
      - C/C++ and PL/I local variables, COBOL LOCAL-STORAGE
    - Must use STACK(,,BELOW) when running ALL31(OFF)

Key Run-Time Options - Tuning

- RPTSTG(option)
  - OFF Storage report not requested
  - ON Generates a report of stack/heap usage
    - including recommended settings
  - Caution:
    - Use only for application tuning. Do not make RPTSTG(ON) system wide default due to significant performance impact.

Consider CICS TS dynamic storage tuning as an alternative.

Key Run-Time Options - Diagnostics

- DYNDUMP(hlq,U4039 Behavior,U40xx Behavior)
  - hlq – may be user-specified, or:
    - *USERID | *USERID.hlq
    - *TSOPREFIX | *TSOPREFIX.hlq
  - U4039 Behavior - with TERMTHDACT(UADUMP/UAONLY/UATRACE)
    - NODYNAMIC – Do not create IPCS-readable dump (default)
    - DYNAMIC – Create IPCS-readable dump if no other dump DD name
    - FORCE – Create IPCS-readable dump instead of other dumps
    - BOTH – Create IPCS-readable dump in addition to other dumps
  - U40xx Behavior – non-U4039 dumps
    - TDUMP – Create IPCS-readable dump (default)
    - NOTDUMP – Do not create IPCS-readable dump

Key Run-Time Options - Diagnostics

- HEAPCHK(ON|OFF, frequency, delay, level, call-depth, num-entries, pool-num)
  - OFF — Normal processing
  - ON — Checks HEAP structures on get/free
  - frequency — How often the HEAP is checked
  - delay — Number of get/free before starting
  - level — Number of calls to be displayed in Heap Storage Diagnostic Report
  - call-depth — Number of calls to be displayed for HEAPPOOLS Serviceability
  - num-entries — Number of entries to be recorded in the heap pool trace table for the main user heap
  - pool-num — ID of the heap pool to be traced

Key Run-Time Options - Diagnostics

- HEAPCHK(ON|OFF, frequency, delay, level, call-depth , num-entries, pool-num) *(continued)*
  - Caution:
    - Use only for application tuning/diagnostics.
      Do not make HEAPCHK(ON) system wide default due to serious performance impact.
  - Notes:
    - To generate only Heap Storage Diagnostic Report use, e.g.
      - HEAPCHK(ON,0,0,10,0)
    - To activate only HEAPPOOLS Serviceability use, e.g.
      - HEAPCHK(ON,0,0,0,5)

Key Run-Time Options - Diagnostics

- STORAGE(getheap, freeheap, stack, reserve)
  - getheap — One byte value used to initialize every heap allocation
  - freeheap — One byte value used to initialize every heap free
  - stack — One byte value used to initialize every stack allocation
  - reserve — Amount of space to reserve for out of storage condition processing

Key Run-Time Options - Diagnostics

- STORAGE(getheap, freeheap, stack, reserve) *(continued)*

Notes:

- STORAGE(AA,EE,,) useful for debugging
  - When HEAPCHK(ON), free elements are checked to ensure they contain the freeheap value
- STORAGE(00,,,) is equivalent to COBOL WSCLEAR
- STORAGE(,,00,) vs. STORAGE(,,CLEAR,)
  - 00 is very expensive (especially for C/C++)
  - CLEAR sets to binary zeros the unused portion of the initial stack segment just prior to the “main” getting control

Key Run-Time Options - Diagnostics

- TERMTHDACT(option)
  - QUIET Messages off, no dump
  - MSG Messages only, no dump
  - TRACE CEEDUMP with traceback only
  - DUMP CEEDUMP
  - UADUMP CEEDUMP, optional system dump
  - UAONLY System dump only, no CEEDUMP
  - UATRACE System dump and traceback

  - Notes:
    - SYSMDUMP DD card required for system dump (unless DYNDUMP is being used)

Key Run-Time Options - Diagnostics

- TRAP(option)
  - ON,SPIE — Condition handling enabled
  - ON,NOSPIE — Allows user applications to have their own SPIE routine, Language Environment condition handling will take place via the ESTAE
  - OFF — Condition handling disabled, some functionality not available **(AVOID)**
  - Notes:
    - TRAP(ON,SPIE) highly recommended for normal processing

Key Run-Time Options - Diagnostics

- RPTOPTS(option)
  - OFF — Options report not requested
  - ON — Generate a report of all current options (upon successful termination)
  - Notes:
    - Automatically included in CEEDUMP

Other Good sessions

What's New in LE for z/OS	Mon	11:00AM
What's New in Enterprise PL/I V4R1 and C/C++ V1.12	Mon	12:15PM
Full Speed Ahead with COBOL into the Future	Mon	1:30PM
REXX Language Coding Techniques	Tues	12:15PM
User Experience: Writing a Web-enabled CICS/COBOL Program	Wed	8:00AM
COBOL Performance – Myths and Realities	Wed	11:00AM
Language Environment Futures Workshop/AMODE 64 Discussion	Wed	4:30PM
An Introduction to using REXX with Language Environment	Thu	1:30PM
Look What I Found Under the Bar!	Thu	3:00PM
Heap Damage, Is Your Insurance Up-to-Date?	Fri	8:00AM
Introducing LE Callable Services, plus a User's View of Why and How You Should Exploit Them in Your Applications	Fri	9:30AM
LE Crime Scene Investigation - Finding debugging clues in LE dumps	Fri	11:00AM


LE
PL/I
C/C++
CEL
COBOL
Fortran

Appendix

- Supported Releases
- Compilers Compatible With LE
- Compilers That Require LE
- The Life of a Module
- Setting Run-time Options

Supported Releases

Supported Release Level	FMID	Support Withdrawn
z/OS V1.10	HLE7750	9/30/2011
z/OS V1.11	HLE7760	9/2012*
z/OS V1.12	HLE7770	
z/OS V1.13	HLE7780	

* Indicates Projected Date

Compilers Compatible with LE

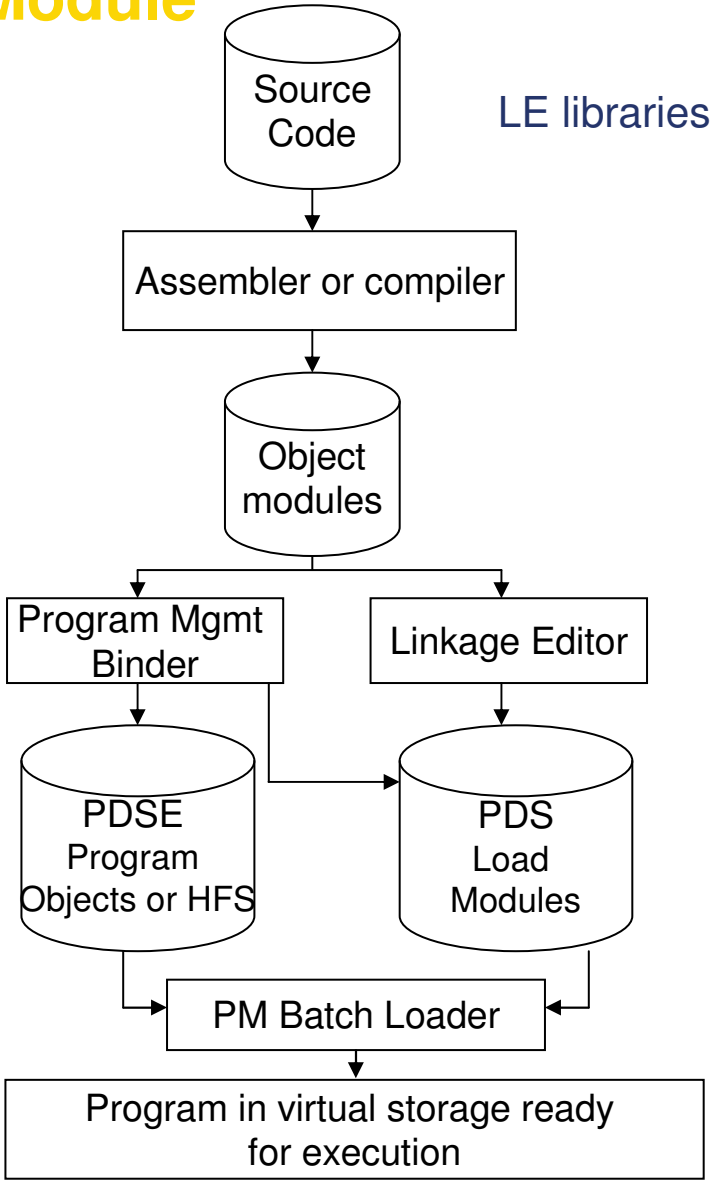
Object modules compiled with the following compilers **will run with LE without having to be re-linked or if linked with LE do not need to be recompiled:**

- C/370 Versions 1 and 2
- OS/VS COBOL Release 2
- VS COBOL II Release 3 or later
- OS PL/I Version 1 Release 3 (object modules), Version 1 Release 5.1 and Version 2, all releases (load modules)
- VS FORTRAN Versions 1 and 2 (MVS only)
- FORTRAN IV H Extended (MVS only)
- FORTRAN IV G1 (MVS only) for OS/390 VS FORTRAN and FORTRAN IV (in compatibility mode)

Compilers that Require LE

z/OS XL C/C++
OS/390 C/C++
C/C++ Compiler for MVS/ESA(TM)
AD/Cycle® C/370(TM) Compiler
VisualAge for Java, Enterprise Edition for OS/390
Enterprise COBOL for z/OS
Enterprise COBOL for z/OS and OS/390
COBOL for OS/390 & VM
COBOL for MVS & VM (formerly COBOL/370)
Enterprise PL/I for z/OS
Enterprise PL/I for z/OS and OS/390
VisualAge PL/I for OS/390
PL/I for MVS & VM
AD/Cycle PL/I for MVS & VM
VS FORTRAN and FORTRAN IV (in compatibility mode)

The Life of a Module



LE libraries that may get involved:

SCEEMAC, SCEEH.*

SCEELKED, SCEELKEX, SCEEOBJ, SCEECPP, SCEELIB, SCEEBND2

SCEERUN, SCEERUN2, SCEELPA, SCEECICS

Setting Run-Time Options

- Installation defaults (CEEDOPT/CEECOPT/CELQDOPT)
  - Also referred to as system-wide defaults
  - SMP/E USERMOD to Language Environment modules
  - All options must be specified

```
CEEDOPT  CSECT                                                  00110000
CEEDOPT  AMODE ANY                                              00120000
CEEDOPT  RMODE ANY                                              00130000
         CEEXOPT ABPERC=((NONE),OVR),                          X00140000
               ABTERMENC=((ABEND),OVR),                        X00150000
               AIXBLD=((OFF),OVR),                             X00160000
               ALL31=((ON),OVR),                               X00170000
               ANYHEAP=((16K,8K,ANYWHERE,FREE),OVR),           X00180000
               BELOWHEAP=((8K,4K,FREE),OVR),                   X00190000
```

Setting Run-Time Options

- System defaults
  - Options may be specified in a PARMLIB member
    - CEEPRMxx
  - Options may be specified with an operator command
    - SETCEE
  - Reduces the need to maintain USERMODs for CEEDOPT/CEECOPT/CELQDOPT

Setting Run-Time Options

- System defaults *(continued)*
  - Specifying options in PARMLIB member
    - Member name CEEPRMxx
      - Member(s) specified at IPL time using CEE=xx via IEASYSxx or at the system parameters prompt
      - Can by dynamically changed via SET CEE=yy command
    - Options specified in groups

```
CEEDOPT(ABPERC(NONE) ALL31(ON)
        rptopts(on) )    /* Options report */

CEECOPT(anyheap(4k,4080,anywhere,free))

CEEDOPT(ALL31(OFF) ) /* Turn off this option */
```

Setting Run-Time Options

- System defaults *(continued)*
  - Using the SETCEE system command
    - Overrides the current system defaults
    - Usage
      - Specify one group per command
      - Up to 126 characters
      - Example:

```
SETCEE ceedopt,rptstg(on),rptopts(on)
```

Setting Run-Time Options

- System defaults *(continued)*
  - Displaying the system defaults
    - D CEE displays the active members

```
d cee
CEE3744I 17.57.31 DISPLAY
CEE=(JM)
```

    - D CEE,groupname displays the options for a particular group

```
d cee,ceedopt
CEE3745I 17.59.44 DISPLAY CEEDOPT
CEE=(01)
LAST WHERE SET  OPTION
------------------------------------------
CEEPRM01        ENVAR("testing=roger","verify=1 2 3")
CEEPRM01        HEAP(4194304,5242880,ANYWHERE,KEEP,
           16384,16384)
CEEPRM01        PROFILE(OFF,"XXX")
CEEPRM01        RPTOPTS(ON)
```

Setting Run-Time Options

- Region Level Overrides (CEEROPT)
  - CICS TS and LRR users (e.g. IMS) only (pre-z/OS V1.10)
  - Batch users (via CEEROPT/CELQROPT CEEPRMxx keyword) (V1.10)
  - Separate load module dynamically loaded at run-time during region initialization
    - SCEESAMP(CEEWROPT)
    - Must be found in search order, such as STEPLIB for IMS and batch, or DFHRPL for CICS TS
  - Specify only those options you wish to change

```
CEEROPT  CSECT                                              00110000
CEEROPT  AMODE ANY                                          00120000
CEEROPT  RMODE ANY                                          00130000
         CEEXOPT ALL31=((ON),OVR),                         X00170000
               STORAGE=((00,NONE,NONE,0K),OVR)              00210000
         END
```

Setting Run-Time Options

- Region Level Overrides (CEEROPT) *(continued)*
  - Certain options can be overridden dynamically in CICS TS region via the CLER transaction
    - ALL31
    - CBLPSHPOP
    - CHECK
    - INFOMSGFILTER
    - RPTOPTS
    - RPTSTG
    - TERMTHDACT
    - TRAP

Setting Run-Time Options

- Application Level Overrides (CEEUOPT/CELQUOPT)
  - CSECT linked with the application
    - SCEESAMP(CEEWUOPT/CEEWQUOP)
  - Specify only those options you wish to change

```
CEEUOPT  CSECT                                                  00110000
CEEUOPT  AMODE ANY                                              00120000
CEEUOPT  RMODE ANY                                              00130000
         CEEXOPT HEAP=(10M,10M,ANYWHERE,FREE),                 X00180000
               STACK=(1M,1M,ANYWHERE,KEEP)                      00250000
         END
```

Setting Run-Time Options

- Programmer Overrides
  - Compiled into program
    - #pragma runopts for C/C++

```
#pragma runopts(ALL31(ON),ERRCOUNT(0),\
          STACK(2M,1M,ANYWHERE,KEEP),\
          HEAP(1M,500K,ANYWHERE,KEEP))
```

    - PLIXOPT for PL/I

```
DCL PLIXOPT CHAR(140) VAR  INIT('ALL31(ON)
          ERRCOUNT(0)  STACK(2M,1M,ANYWHERE,KEEP)
HEAP(1M,500K,ANYWHERE,KEEP)')  STATIC   EXTERNAL;
```

    - not available for COBOL
  - Internally generates CEEUOPT/CELQUOPT

Setting Run-Time Options

- Program Invocation Overrides
  - In UNIX System Services shell (case sensitive)
    - export _CEE_RUNOPTS=‘run-time options’
  - In batch, on EXEC card
    - COBOL (with CBLOPTS(ON))
      - PARM='program arguments/run-time options'
    - C/C++, PL/I, FORTRAN, Language Environment-conforming Assembler
      - PARM='run-time options/program arguments'
    - First program must be Language Environment-conforming
    - The slash is required to delineate the run-time options, even when no program arguments.
    - Note that PARM= is limited to 100 characters

Setting Run-Time Options

- DD:CEEOPTS Overrides
  - Optional data set in which run-time options may be specified
  - Allows up to 3K characters
  - Allows run-time options to be passed to non-Language Environment conforming main routines

```
//MYAPPL01 EXEC
PROG=MYPRG,PARM='RPTOPTS(ON)/'
//CEEOPTS  DD *
* THESE ARE MY OPTIONS:
ALL31(ON), HEAP(64K),
ENVAR("JOHN=MONTI"),
TERMTHDACT(UADUMP)
/*
```