Language Environment
Runtime Application Migration Guide

Version 2 Release 1
Before using this information and the product it supports, read the information in "Notices" on page 31.
## Tables

1. How to use z/OS Language Environment publications .................................. viii
2. ILC compatibility exceptions ................................................................. 12
3. COBOL compatibility exceptions ......................................................... 13
4. Fortran compatibility exceptions ......................................................... 13
5. PL/I compatibility exceptions ......................................................... 14
6. C and Language Environment runtime options ............................... 17
7. COBOL and Language Environment runtime options ............................... 18
8. Fortran and Language Environment runtime options ............................... 20
9. PL/I and Language Environment runtime options ............................... 21
IBM® z/OS Language Environment (also called Language Environment) provides common services and language-specific routines in a single runtime environment for C, C++, COBOL, Fortran (z/OS only; no support for z/OS UNIX System Services or CICS®), PL/I, and assembler applications. It offers consistent and predictable results for language applications, independent of the language in which they are written.

Language Environment is the prerequisite runtime environment for applications generated with the following IBM compiler products:
- z/OS XL C/C++ (feature of z/OS)
- z/OS® C/C++
- OS/390® C/C++
- C/C++ for MVS/ESA
- C/C++ for z/VM®
- XL C/C++ for z/VM
- AD/Cycle C/370™
- 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
- PL/I for MVS & VM (formerly PL/I MVS™ & VM)
- VS FORTRAN and FORTRAN IV (in compatibility mode)

Although not all compilers listed are currently supported, Language Environment supports the compiled objects that they created.

Language Environment supports, but is not required for, an interactive debug tool for debugging applications in your native z/OS environment.

Debug Tool is also available as a standalone product. Debug Tool Utilities and Advanced Functions is also available. For more information about Debug Tool for z/OS, see the Debug Tool for z/OS home page (http://www.ibm.com/software/products/us/en/debugtool).

Language Environment supports, but is not required for, VS FORTRAN Version 2 compiled code (z/OS only).

Language Environment consists of the common execution library (CEL) and the run-time libraries for C/C++, COBOL, Fortran, and PL/I.

For more information on VisualAge for Java, Enterprise Edition for OS/390, program number 5655-JAV, see the product documentation.
This book provides an overview of the steps z/OS customers must take to migrate applications for use with z/OS Language Environment. These customers may not necessarily be migrating to a new language compiler.

This book is written for application developers. Familiarity with the runtime libraries of the different languages, and an understanding of the basics of linking and running applications, are assumed.

The information in this book will not provide a comprehensive guide to the migration process; rather, it is designed to help you create a broad migration strategy. This book will help you identify which modules can be migrated first, and which will require relinking or recompiling. It also explains how to use Language Environment runtime options to achieve behavior that is compatible with your old modules. For more detailed information about migration topics such as upgrading source code and load module compatibility, see one of the following documents or Web sites:

- z/OS XL C/C++ Compiler and Runtime Migration Guide for the Application Programmer
- IBM C for VM/ESA Compiler and Run-Time Migration Guide
- Fortran Run-Time Migration Guide
- The IBM Enterprise PL/I for z/OS library (http://www.ibm.com/support/docview.wss?uid=swg27036735)
- The Enterprise COBOL for z/OS library (http://www-01.ibm.com/support/docview.wss?uid=swg27036733)

### Using your documentation

The publications provided with Language Environment are designed to help you:
- Manage the runtime environment for applications generated with a Language Environment-conforming compiler.
- Write applications that use the Language Environment callable services.
- Develop interlanguage communication applications.
- Customize Language Environment.
- Debug problems in applications that run with Language Environment.
- Migrate your high-level language applications to Language Environment.

Language programming information is provided in the supported high-level language programming manuals, which provide language definition, library function syntax and semantics, and programming guidance information.

Each publication helps you perform different tasks, some of which are listed in Table 1.

<table>
<thead>
<tr>
<th>To ...</th>
<th>Use ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evaluate Language Environment</td>
<td>z/OS Language Environment Concepts Guide</td>
</tr>
<tr>
<td>Plan for Language Environment</td>
<td>z/OS Language Environment Concepts Guide</td>
</tr>
<tr>
<td>Install Language Environment</td>
<td>z/OS Program Directory</td>
</tr>
<tr>
<td>To ...</td>
<td>Use ...</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Customize Language Environment</td>
<td>z/OS Language Environment Customization</td>
</tr>
<tr>
<td>Understand Language Environment</td>
<td>z/OS Language Environment Concepts Guide</td>
</tr>
<tr>
<td>program models and concepts</td>
<td>z/OS Language Environment Programming Guide</td>
</tr>
<tr>
<td></td>
<td>z/OS Language Environment Programming Guide for 64-bit Virtual Addressing Mode</td>
</tr>
<tr>
<td>Find syntax for Language Environment</td>
<td>z/OS Language Environment Programming Reference</td>
</tr>
<tr>
<td>runtime options and callable services</td>
<td></td>
</tr>
<tr>
<td>Develop applications that run with</td>
<td>z/OS Language Environment Programming Guide and your language programming guide</td>
</tr>
<tr>
<td>Language Environment</td>
<td></td>
</tr>
<tr>
<td>Debug applications that run with</td>
<td>z/OS Language Environment Debugging Guide</td>
</tr>
<tr>
<td>Language Environment, diagnose</td>
<td></td>
</tr>
<tr>
<td>problems with Language Environment</td>
<td></td>
</tr>
<tr>
<td>Get details on runtime messages</td>
<td>z/OS Language Environment Runtime Messages</td>
</tr>
<tr>
<td>Develop interlanguage communication</td>
<td>z/OS Language Environment Writing Interlanguage Communication Applications and your language programming guide</td>
</tr>
<tr>
<td>(ILC) applications</td>
<td></td>
</tr>
<tr>
<td>Migrate applications to Language</td>
<td>z/OS Language Environment Runtime Application Migration Guide</td>
</tr>
<tr>
<td>Environment</td>
<td>and the migration guide for each Language Environment-enabled language</td>
</tr>
</tbody>
</table>

**z/OS information**

This information explains how z/OS references information in other documents and on the web.

When possible, this information uses cross document links that go directly to the topic in reference using shortened versions of the document title. For complete titles and order numbers of the documents for all products that are part of z/OS, see [z/OS Information Roadmap](http://www.ibm.com/systems/z/os/zos/bkserv/).

To find the complete z/OS library, including the z/OS Information Center, see [z/OS Internet Library](http://www.ibm.com/systems/z/os/zos/bkserv/).
How to send your comments to IBM

We appreciate your input on this publication. Feel free to comment on the clarity, accuracy, and completeness of the information or provide any other feedback that you have.

Use one of the following methods to send your comments:
1. Send an email to mhvrdfs@us.ibm.com.
2. Send an email from the "Contact us" web page for z/OS (http://www.ibm.com/systems/z/os/zos/webqs.html).
3. Mail the comments to the following address:
   IBM Corporation
   Attention: MHVRCFS Reader Comments
   Department H6MA, Building 707
   2455 South Road
   Poughkeepsie, NY 12601-5400
   US
4. Fax the comments to us, as follows:
   From the United States and Canada: 1+845+432-9405
   From all other countries: Your international access code +1+845+432-9405

Include the following information:
• Your name and address.
• Your email address.
• Your telephone or fax number.
• The publication title and order number:
  z/OS V2R1.0 Language Environment Runtime Application Migration Guide
  GA32-0912-00
• The topic and page number that is related to your comment.
• The text of your comment.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute the comments in any way appropriate without incurring any obligation to you.

IBM or any other organizations use the personal information that you supply to contact you only about the issues that you submit.

If you have a technical problem

Do not use the feedback methods that are listed for sending comments. Instead, take one of the following actions:
• Contact your IBM service representative.
• Call IBM technical support.
• Visit the IBM Support Portal at z/OS support page (http://www.ibm.com/systems/z/support/).
z/OS Version 2 Release 1 summary of changes

See the following publications for all enhancements to z/OS Version 2 Release 1 (V2R1):

- z/OS Migration
- z/OS Planning for Installation
- z/OS Summary of Message and Interface Changes
- z/OS Introduction and Release Guide
Chapter 1. Planning to migrate to Language Environment

This topic provides a checklist to help you plan the migration of your applications to the Language Environment runtime environment for the first time.

For more detailed information about migration considerations, see Chapter 3, “Migrating from other runtime environments,” on page 11. If you are migrating from a previous release of Language Environment, you should review the information in Chapter 2, “Migrating from another Language Environment release,” on page 5.

Checklist for migration

Each task in the following checklist is recommended; you should perform each task in the order shown.

1. Learn about Language Environment.
   Ensure that you and other application programmers who will be involved in the migration effort are familiar with the features of Language Environment and the differences between your current runtime environment and the Language Environment runtime environment. You can get information about Language Environment from publications such as:
   • z/OS Language Environment Customization
   • z/OS Language Environment Concepts Guide

2. Take an inventory of the applications and vendor products you intend to run with Language Environment.
   • C, C++, COBOL, Fortran, PL/I, or Assembler programs
     For each program you intend to move to the Language Environment runtime environment, obtain the following information:
     – Version and release of the compiler that generated the program
     – Which COBOL programs were compiled with RES and which with NORES
     – Runtime options used and how they were specified
     – Which PL/I programs use the shared library and which ones do not
     – Which programs call, or are called by, assembler programs
     – Which applications contain interlanguage communication (ILC)
     – Which programs are used with CICS, IMS™, DB2®, or other subsystems
     – Control statements used
     – Frequency and types of abends
     – Test cases required and available
     – Amount of storage used
     – Frequency of execution of reusable or common modules
     – Program execution time (processor (CPU) and elapsed)
   • Vendor tools, packages, and products
     – Ensure that all vendor tools, packages, and products run with Language Environment; any source code for the packages must also be compatible with your Language Environment-conforming compiler.
Planning to migrate

- Ensure that any vendor code generators generate code that is compatible with your Language Environment-conforming compiler.
- Ensure that vendor development tools and debuggers will not issue their own ESPIE or ESTAE, as Language Environment must get control first.

3. Prioritize programs.

Determine the effort required to migrate each program and the order in which you will migrate them. Each program will require some level of effort to migrate, ranging from minimal testing to a code rewrite. Using the information from your inventory analysis, determine if each program:

- Requires minimum, moderate, or extensive testing
- Runs with Language Environment without change
- Requires relinking with Language Environment
- Must be recompiled with a Language Environment-conforming compiler, without change to the source code
- Requires changes to the source code
- Does not run with Language Environment

After you have determined the effort required to migrate each load module, list your programs in the order you want to move them to Language Environment. You should consider the importance of each program and how often it is used. You should migrate applications that contain ILC after you have migrated any applications that contain only C, C++, COBOL, Fortran, or PL/I. (An application that contains assembler, but is otherwise created from one language, is not considered an ILC application in this information.) For information about compatibility considerations for ILC applications, see "Migrating ILC applications to Language Environment" on page 12.


Perform the following tasks, which can be done concurrently:

- Change default runtime options as appropriate.
  
  To ensure that the Language Environment runtime results are compatible with your current runtime results, you will need to change some of the default settings for the runtime options. For a list of recommended settings, see z/OS Language Environment Customization.

- Assess storage requirements.
  
  Storage requirements may be larger for Language Environment than for your current runtime environment. During conversion, you might need DASD for the Language Environment runtime library and for the runtime library that you are currently using. For information about Language Environment DASD requirements, see z/OS Program Directory.

  Virtual storage requirements for placing library routines above or below the 16M line may also increase, depending on which Language Environment storage options you specify. For recommended settings, see z/OS Language Environment Customization.

- Determine how to phase-in the Language Environment runtime environment using a STEPLIB approach or by adding Language Environment to the LNKLST.

  Using the STEPLIB approach, you can gradually phase-in the Language Environment runtime environment. When you use STEPLIB statements to specify the Language Environment runtime environment, you can phase-in one region (CICS or IMS), batch (group of applications), or user (TSO) at a time. Although using STEPLIB means changing JCL, a gradual conversion can be easier than moving all of your applications at one time.
Planning to migrate

When you add Language Environment to the LNKLST, it is available to all of your applications. Ensure all applications are functioning correctly with Language Environment before adding Language Environment to your LNKLST. You might consider temporarily adding Language Environment to the LNKLST until you have confirmed the applications work as intended.

5. **Set up a regression testing procedure.**
   To ensure that the Language Environment runtime results are compatible with your current runtime results, you will need to perform regression tests on all the programs you migrate. Run your applications in parallel with your current runtime environment and with the Language Environment runtime environment to confirm that the results are the same. You can temporarily add Language Environment to the LINKLST to accomplish this. When your applications are running with Language Environment in a test environment, you should take performance measurements, especially on any time-critical or response-critical applications.

6. **Move applications into procedure.**
   When your testing shows the entire application (or group of applications, if running more than one application in an IMS region or under TSO) runs as expected, you can move the entire unit over to production use. However, if an unexpected error occurs, you may need to perform one of the following steps:
   - On z/OS systems, run the previous version of your application as a substitute.
   - Under DB2, CICS, and IMS, return to the last commit point and then continue processing from that point using the previous version of the program. For DB2, use an SQL ROLLBACK WORK statement.
   - For batch applications, use the backup and restore facilities at your site to recover.

   After you move your applications to production use with the Language Environment runtime environment, monitor your applications to ensure that they continue to work properly. You can then run with the confidence that you had in your previous runtime library.

Planning to link and run with Language Environment

Language Environment provides separate libraries for linking and running applications. The link libraries, of which SCEELKED is one, contain static (resident) routines that are linked with the application and used to resolve external references at link-edit time. The load library, of which SCEERUN is one, contains dynamic routines that are not part of the application and are dynamically loaded at run time. Language Environment callable services and other routines, such as those for initialization and termination, are located in SCEERUN. For a complete list of libraries and which phase of application development they are used in, see [z/OS Language Environment Customization](https://www.ibm.com/downloads/whitepaper/IBM-ZOS-Language-Environment-Customization). You will need to modify the job control statements in your input stream to point directly to SCEELKED and SCEERUN, or to point to the appropriate IBM-supplied cataloged procedures, if your job uses cataloged procedures. See [z/OS Language Environment Programming Guide](https://www.ibm.com/support/knowledgecenter/SSSH75_2.1.0/com.ibm.zos.zos.doc/planltc32.html) for more information about linking and running and using cataloged procedures.

On z/OS systems, you can install reentrant members of the SCEERUN data set in the link pack area (LPA) for faster retrieval. IBM provides a data set called
Planning to migrate

SCEELPA and highly recommends putting this data set in the LPA (or LPALSTxx). This data set contains modules that are reentrant, reside above the line, and are heavily used by z/OS itself.
Chapter 2. Migrating from another Language Environment release

This topic provides information about migrating from one release of Language Environment to another. This topic explains upward compatibility as well as downward compatibility. Please look at any migration considerations listed in this topic for the release you are migrating to as well as any releases you are skipping over.

Language Environment provides general object and load module compatibility for applications that ran with a previous release of Language Environment. All Language Environment-enabled applications that have been linked with a minimum level of Language Environment for MVS and VM 1.3 will continue to run with later releases of Language Environment without the need to relink the application. If you experience any problems (for example, an application that worked with Language Environment for MVS and VM 1.3 no longer works after you install the current release of Language Environment), you should report them to IBM.

Most load modules are compatible with any level of Language Environment that is equivalent to, or higher than, the level used to link-edit them. Similarly, object modules can be link-edited with any level of Language Environment that is equivalent to, or higher than, the level required by the compiler that generated them.

Please see any exceptions in this topic.

Migration actions required for each release

This topic describes common activities and considerations that are typically required when you migrate from one release of Language Environment to another:

1. Update the CICS System Definition (CSD) file using the program definitions in the CEECCSD member and CEECCSDX member found in the SCEESAMP data set. Language Environment may have changed (added or deleted) load modules in this release.

2. Update the Language Environment load modules that were placed in the link pack area (LPA). Sample members found in the SCEESAMP data set, which can be used to move load modules into LPA, should be reviewed with every release migration. See the table “Language Environment sample IEALPAnn or PROGxx members in CEE.SCEESAMP” for the list of sample members and their changed content in z/OS Language Environment Customization.

3. If any Language Environment user exits were used at the previous release and you plan to use them with the new release, they must be re-linked using the new release of Language Environment.

4. In z/OS V1R7, a new parmlib member, CEEPRMxx, was added for Language Environment. You can use it to specify Language Environment runtime options for the system. Operator commands are also provided to allow you to query and update the active runtime options for the system. This simplifies the management of Language Environment options, particularly in multisystem environments. For more information, see z/OS Language Environment Customization.
Migration considerations for Language Environment in z/OS V1R13

There are no runtime application migration considerations for z/OS V1R13.

Migration considerations for Language Environment in z/OS V1R12

Setting runtime options as overrideable or nonoverrideable

**Description:** Before z/OS V1R12, all runtime options specified in a CEEPRMxx parmlib member were overrideable by default. Beginning with z/OS V1R12, you can set runtime options as overrideable or nonoverrideable in the CEEPRMxx parmlib member or with a SETCEE command using the OVR or NONOVR attribute. The ability to specify an option as overrideable or nonoverrideable removes a barrier to using CEEPRMxx.

<table>
<thead>
<tr>
<th>Element or feature</th>
<th>Language Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>When change was introduced:</td>
<td>z/OS V1R12.</td>
</tr>
<tr>
<td>Applies to migration from:</td>
<td>z/OS V1R11 and z/OS V1R10.</td>
</tr>
<tr>
<td>Timing:</td>
<td>Before the first IPL of z/OS V1R12.</td>
</tr>
<tr>
<td>Is the migration action required?</td>
<td>No, but recommended so you can eliminate use of the assembler language USERMODs to specify installation-wide runtime options, and use parmlib member CEEPRMxx instead.</td>
</tr>
</tbody>
</table>

Target system hardware requirements: None.

Target system software requirements: None.

Other system (coexistence or fallback) requirements: None.

Restrictions: None.

System impacts: None.

Related IBM Health Checker for z/OS check: None.

**Steps to take:** Set runtime options as overrideable or nonoverrideable in the CEEPRMxx parmlib member or by issuing the SETCEE command using the OVR or NONOVR attribute.

Now that runtime options can be specified as overrideable or nonoverrideable in a CEEPRMxx parmlib member, and with a SETCEE command, you can eliminate the use of assembler language USERMODs to specify installation-wide runtime options.

Reference information:

- For more information about specifying a CEEPRMxx parmlib member, see [z/OS Language Environment Customization](https://publib.boulder.ibm.com/infocenter/lnceavadoc/v1r0m/topic/omvamnm086b.htm).
- For the updated CEEPRM00 sample parmlib member, which includes every option specified as overrideable, see the CEE.SCEESAMP data set.
Changes to the Language Environment runtime options report when issuing D CEE

Description: Starting in z/OS V1R12, changes are made to the Language Environment runtime options report when a D CEE command is issued. Before z/OS V1R12, the Language Environment runtime options report displayed all suboptions, even if they were not explicitly set for any runtime option that was specified. Starting in z/OS V1R12, when a valid option is specified with a parmlib member or a SETCEE command, only the suboptions specified are displayed when a D CEE command is issued. A comma is displayed as a placeholder for those suboptions not specified.

<table>
<thead>
<tr>
<th>Element or feature</th>
<th>Language Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>When change was introduced:</td>
<td>z/OS V1R12.</td>
</tr>
<tr>
<td>Applies to migration from:</td>
<td>z/OS V1R11 and z/OS V1R10.</td>
</tr>
<tr>
<td>Timing:</td>
<td>Before the first IPL of z/OS V1R12.</td>
</tr>
<tr>
<td>Is the migration action required?</td>
<td>Yes, if you have an application that reads the output of a D C EE command.</td>
</tr>
<tr>
<td>Target system hardware requirements:</td>
<td>None.</td>
</tr>
<tr>
<td>Target system software requirements:</td>
<td>None.</td>
</tr>
<tr>
<td>Other system (coexistence or fallback) requirements:</td>
<td>None.</td>
</tr>
<tr>
<td>Restrictions:</td>
<td>None.</td>
</tr>
<tr>
<td>System impacts:</td>
<td>None.</td>
</tr>
<tr>
<td>Related IBM Health Checker for z/OS check:</td>
<td>None.</td>
</tr>
</tbody>
</table>

Steps to take: Examine any programs that read the output of a D CEE command to ensure compatibility with the updated runtime options report. Commas are now displayed for any suboptions that are not explicitly specified in a parmlib member or with a SETCEE command.

For example, if the following SETCEE command is issued:

```
SETCEE CEEDOPT,ALL31,ANYHEAP(4K),FILETAG(,AUTOTAG)
```

a subsequent D CEE,CEEDOPT command displays the following:

```
CEE3745I 09.32.13 DISPLAY CEEDOPT
NO MEMBERS SPECIFIED
-----------------------------------------------
SETCEE command ALL31()
SETCEE command ANYHEAP(4096,)
SETCEE command FILETAG(,AUTOTAG)
```

Reference information: For more information, see z/OS Language Environment Customization and z/OS MVS System Commands.

Removal of conversion table source code

Description: Starting in z/OS V1R12, the C/C++ runtime library will no longer ship any ucmap source code or genxlt source code for character conversions now being performed by Unicode Services.
Migrating from another Language Environment release

<table>
<thead>
<tr>
<th>Element or feature</th>
<th>Language Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>When change was introduced:</td>
<td>z/OS V1R12.</td>
</tr>
<tr>
<td>Applies to migration from:</td>
<td>z/OS V1R11 and z/OS V1R10.</td>
</tr>
<tr>
<td>Timing:</td>
<td>After the first IPL of z/OS V1R12.</td>
</tr>
<tr>
<td>Is the migration action required?</td>
<td>Yes, if you use the iconv() family of functions to test to a “known conversion result” and experience testcase failures. Also, if you use custom conversion tables replacing those listed in either ucmapt.lst or genxlt.lst.</td>
</tr>
</tbody>
</table>

| Target system hardware requirements: | None. |
| Target system software requirements: | None. |
| Other system (coexistence or fallback) requirements: | None. |
| Restrictions: | None. |
| System impacts: | None. |
| Related IBM Health Checker for z/OS check: | None. |

Steps to take:
• If you use customized conversion tables, you should now generate custom Unicode Services conversion tables.
• If you use the iconv() family of functions testing to a "known conversion result" and experience test case failures, you need to update your expected results to the new conversion results.
• If you want to create custom conversion tables involving any of the CCSIDs related to the conversion table source no longer being shipped, you should now generate custom Unicode Services conversion tables instead of custom Language Environment conversion tables.

The installation prefix:SCEEUMAP data set will no longer be shipped.

The /usr/lib/nls/locale/ucmap HFS directory will no longer be shipped.

Note: The _ICONV_TECHNIQUE environment variable must be set to the same technique search order value used for the customized Unicode Services table in order for the iconv() family of functions to use the customized Unicode Services table. For example, if you want the iconv() family of functions to use a user-defined Unicode Services table with a technique search order of 2, the _ICONV_TECHNIQUE environment variable should be set to 2LMREC.

Reference information: For information on how to generate and use custom Unicode Services conversion tables, see z/OS Unicode Services User’s Guide and Reference.

Changes to the CICS CLER runtime options report

Description: Starting with z/OS V1R12, the Language Environment runtime options report displayed from the CICS CLER transaction is changed. The report is modified to have a wider LAST WHERE SET column to accomodate longer values, such as "Installation Non-overrideable." In addition, the report heading OPTIONS is changed to OPTION to match the other runtime options reports.
## Migrating from another Language Environment release

<table>
<thead>
<tr>
<th>Element or feature</th>
<th>Language Environment.</th>
</tr>
</thead>
<tbody>
<tr>
<td>When change was introduced:</td>
<td>z/OS V1R12.</td>
</tr>
<tr>
<td>Applies to migration from:</td>
<td>z/OS V1R11 and z/OS V1R10.</td>
</tr>
<tr>
<td>Timing:</td>
<td>After the first IPL of z/OS V1R12.</td>
</tr>
<tr>
<td>Is the migration action required?</td>
<td>Yes, if you have an application that reads the output of a CICS CLER transaction.</td>
</tr>
</tbody>
</table>

### Target system hardware requirements: None.

### Target system software requirements: None.

### Other system (coexistence or fallback) requirements: None.

### Restrictions: None.

### System impacts: None.

### Related IBM Health Checker for z/OS check: None.

### Steps to take:

Examine programs that read the output of a CICS CLER transaction to ensure compatibility with the updated CLER runtime options report. The LAST WHERE SET column is now wider and the OPTIONS heading is changed to OPTION. The following is a subset of the new report to show the formatting changes:

<table>
<thead>
<tr>
<th>LAST WHERE SET</th>
<th>OPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Installation default</td>
<td>ABPERC(NONE)</td>
</tr>
<tr>
<td>Installation default</td>
<td>ABTERMENC(ABEND)</td>
</tr>
<tr>
<td>Installation default</td>
<td>NOAIXBLD</td>
</tr>
</tbody>
</table>

### Reference information:

For more information, see [z/OS Language Environment Programming Guide](#) and [z/OS Language Environment Debugging Guide](#).

## Migration considerations for Language Environment in z/OS V1R11

### Changes to the HEAPCHK runtime option

The HEAPCHK runtime option now has 4 additional suboptions. Users who use the CEEDOPT, CEECOPT and CELQDOPT usermods to set their installation default runtime options must make a change. Users who use CEEPRMxx to set their system default runtime options rather than the user mods are unaffected.

### Steps to take if you use the usermods:

1. Consider using the CEEPRMxx parmlib member.
2. Compare your existing source for the installation-wide runtime options CSECT, CEEDOPT (non-CICS environment), CEECOPT (CICS environment), or CELQDOPT (AMODE 64 environment) with the new samples in the hlq.SCEESAMP data set to determine whether you need to change your defaults. If changes are necessary, you must make them and apply the corresponding usermods.
3. Understand the new HEAPCHK runtime option suboptions and their default values.
4. Determine if the default values are acceptable for your installation and adjust if needed.
Migrating from another Language Environment release

For more information about specifying the HEAPCHK runtime option, see z/OS Language Environment Customization or z/OS Language Environment Programming Reference.

Changes to binary and decimal floating-point support in the CICS environment

Certain Binary and Decimal Floating-Point Exceptions that were previously reported with a CEE3207S message are now reported with the following messages: CEE3216S, CEE3217S, CEE3218S, CEE3219S, CEE3220S, CEE3221S, CEE3222S, CEE3223S, CEE3224S, CEE3225S, CEE3226S, CEE3227S, CEE3228S, CEE3229S, CEE3231S, CEE3232S, or CEE3233S. These messages are the same ones used in non-CICS environments for Floating-Point exceptions.

The floating-point control register (FPC), floating-point registers 1,3,5,7,8-15, access registers (ARs), and high registers (HRs) are now saved and restored when applications are resumed after program checks and ABENDs. Any changes to these registers made by user condition handlers or signal catchers may be ignored when the registers are restored and the application is resumed.

The sample USRHDLR program, CEEWUCHA, has been changed to check for the new floating program check conditions 3216-3229 and 3231-3233. Any customized versions of CEEWUCHA that are in use may need to be updated.
Chapter 3. Migrating from other runtime environments

This topic describes, in general, the compatibility of Language Environment with previous runtime libraries. It also describes what you must do to migrate different object and load modules to Language Environment.

Note: This publication does not describe all migration considerations. For a detailed description of migration considerations, see the appropriate language migration guide listed in the following topic.

Compatibility with previous runtime libraries

With certain exceptions, Language Environment provides object and load module compatibility for applications that are generated with the following pre-Language Environment IBM language products. Load modules that are created with these compilers and link-edited with their associated runtime libraries run compatibly with Language Environment without relinking. Also, object modules created with these compilers can be linked and run with Language Environment without recompiling.

- 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)

The following topics contain some basic information to help you determine if your applications will run compatibly with Language Environment. For more detailed information about compatibility, see one of the following migration guides:

- IBM C for VM/ESA Compiler and Run-Time Migration Guide
- The appropriate version of the COBOL migration guide in the COBOL library at [Enterprise COBOL for z/OS library](http://www-01.ibm.com/support/docview.wss?uid=swg27036733)
- VisualAge PL/I for OS/390 Compiler and Run-Time Migration Guide
- Fortran Run-Time Migration Guide
Table 2 lists some of the compatibility exceptions you should consider when migrating ILC applications to Language Environment.

**Table 2. ILC compatibility exceptions**

<table>
<thead>
<tr>
<th>To migrate:</th>
<th>You need to:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load modules that contain OS/VS COBOL, with calls to, or from, OS PL/I</td>
<td>Upgrade the COBOL source code and compile with Enterprise COBOL for z/OS or COBOL for OS/390 &amp; VM.</td>
</tr>
<tr>
<td>Load modules that contain VS COBOL II Version 1 Release 3 or later, with calls to, or from, OS PL/I</td>
<td>Relink with Language Environment. However, if you link your VS COBOL II-OS PL/I ILC applications with the migration tool provided by OS PL/I Version 2 Release 3, you will not need to relink your applications. The PTF numbers for the migration aid are UN76954 and UN76955. For information about the migration tool, see the Enterprise COBOL for z/OS library (<a href="http://www-01.ibm.com/support/docview.wss?uid=swg27036733">http://www-01.ibm.com/support/docview.wss?uid=swg27036733</a>) or the IBM Enterprise PL/I for z/OS library (<a href="http://www.ibm.com/support/docview.wss?uid=swg27036735">http://www.ibm.com/support/docview.wss?uid=swg27036735</a>).</td>
</tr>
<tr>
<td>C/370 Version 2 Release 2 (V2R2) load modules that contain calls to, or from, VS COBOL II, COBOL/370, or COBOL for MVS &amp; VM programs</td>
<td>Apply the PTF associated with APAR PN74931, which allows you to relink C/370 V2R2 load modules with the C/370 V2R2 library and run with Language Environment or the C/370 V2R2 library.</td>
</tr>
</tbody>
</table>

See [z/OS XL C/C++ Compiler and Runtime Migration Guide for the Application Programmer](http://www.ibm.com/support/docview.wss?uid=swg27036733) or [IBM C for VM/ESA Compiler and Run-Time Migration Guide](http://www.ibm.com/support/docview.wss?uid=swg27036733) for detailed instructions on how to relink C-COBOL ILC applications. (You do not need to relink PL/I-C ILC applications.)

See the [IBM Enterprise PL/I for z/OS library](http://www.ibm.com/support/docview.wss?uid=swg27036735) for instructions on how to relink PL/I-COBOL ILC applications, and for information about a migration aid that helps migrate OS PL/I-VS COBOL II ILC applications.

Migrating C routines to Language Environment

Generally, you can directly migrate most C/370 Version 1 or Version 2 applications to any release of Language Environment. However, you must use the Language Environment libraries to relink an application if a load module contains one of the following items:

- ILC calls to, and from, Fortran or in some cases COBOL (see Table 2 on page 12)
- Debugging information (that is, they are compiled with the TEST option)
- System Programming C Facility (SPC) load modules that contain dynamic C/370 library functions

For detailed information about migrating your C applications, see z/OS XL C/C++ Compiler and Runtime Migration Guide for the Application Programmer.

Migrating COBOL programs to Language Environment

Table 3 contains a subset of COBOL compatibility exceptions.

<table>
<thead>
<tr>
<th>To migrate:</th>
<th>You need to:</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS/VS COBOL programs mixed with assembler under non-CICS</td>
<td>Run in a single run unit (SVC LINK is not allowed).</td>
</tr>
<tr>
<td>OS/VS COBOL programs that use ILC with PL/I</td>
<td>Upgrade the COBOL source code to Enterprise COBOL for z/OS or COBOL for OS/390 &amp; VM.</td>
</tr>
<tr>
<td>OS/VS COBOL programs that use ILC with FORTRAN</td>
<td>Upgrade the COBOL source code to Enterprise COBOL for z/OS or COBOL for OS/390 &amp; VM.</td>
</tr>
<tr>
<td>VS COBOL II programs that use ILC with C or PL/I</td>
<td>See Table 2 on page 12 for information about migration aids for each language.</td>
</tr>
</tbody>
</table>

Recommendation: Do not install more than one library for a language in the LNKLST or LPALST. You should not concatenate pre-Language Environment runtime libraries in the LNKLST.

For detailed migration information about LNKLST concatenation and COBOL, see the appropriate version of the COBOL migration guide in the Enterprise COBOL for z/OS library (http://www-01.ibm.com/support/docview.wss?uid=swg27036733).

Migrating Fortran routines to Language Environment

Table 4 lists some compatibility exceptions to consider when migrating Fortran applications to Language Environment. For more information, see Fortran Run-Time Migration Guide.

<table>
<thead>
<tr>
<th>To migrate:</th>
<th>You need to:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object modules compiled with VS FORTRAN Version 1 Release 2.0 or earlier and are either programs or subprograms that receive character arguments or pass character arguments to subprograms</td>
<td>Recompile with VS FORTRAN Version 2 and run under Language Environment.</td>
</tr>
</tbody>
</table>
### Migrating other runtime environments

**Table 4. Fortran compatibility exceptions (continued)**

<table>
<thead>
<tr>
<th>To migrate:</th>
<th>You need to:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object modules compiled with VS FORTRAN Version 2 Release 5 or 6 that contain parallel constructs, use the PARALLEL compile-time option, or invoke PEORIG, PEPOST, PEWAIT, PETERM, PLCOND, PLFREE, PLLOCK, PLORIG, or PLTERM</td>
<td>Continue to link-edit and run under VS FORTRAN Version 2. These object modules cannot run under Language Environment.</td>
</tr>
</tbody>
</table>
| Object modules compiled with VS FORTRAN Version 2 Release 5 or 6 using the EC compiler option | Perform one of the following actions, as these object modules cannot run under Language Environment:  
  - Continue to link-edit and run under VS FORTRAN Version 2 Release 5 or 6, or  
  - Remove the EC option from your source, if possible, then recompile and run with Language Environment. |
| Object modules with calls to DVCHK or OVERFL services                       | Remove the calls, change the logic of the program and recompile with VS FORTRAN Version 2.            |
| Object modules that have dependences on product internals                  | Remove the dependencies, change the logic of the program and recompile with VS FORTRAN Version 2. |
| Object modules that have misaligned vector operands                        | Ensure that all vector operands are properly aligned and recompile with VS FORTRAN Version 2. |
| Object modules that use static debug                                       | Remove the debug packets and recompile with VS FORTRAN Version 2.                                     |
| Load modules that contain Fortran with calls to, or from, any other language | See [Table 2 on page 12](#) for instructions.                                                        |

### Migrating PL/I routines to Language Environment


**Table 5. PL/I compatibility exceptions**

<table>
<thead>
<tr>
<th>To migrate:</th>
<th>You need to:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object modules created with OS PL/I Version 1 Release 1 through Version 1 Release 2.3 compilers</td>
<td>Recompile with Enterprise PL/I, PL/I for MVS &amp; VM or with the OS PL/I Version 2 compiler.</td>
</tr>
<tr>
<td>Load modules created with OS PL/I Version 1 Releases 3 through 5.</td>
<td>Relink with Language Environment or with OS PL/I Version 2.</td>
</tr>
</tbody>
</table>
| Load modules created with OS PL/I Version 1 Release 5.1.                   | Apply the IBM-supplied program fix (ZAP) before running the following types of OS PL/I V1 R5.1 load modules:  
  - Main load modules for MVS non-shared library, non-CICS, nonmultitasking |
Migrating other runtime environments

Table 5. PL/I compatibility exceptions (continued)

<table>
<thead>
<tr>
<th>To migrate:</th>
<th>You need to:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load modules that use the OS PL/I shared library</td>
<td>Relink or recompile load modules from OS PL/I Version 1 Releases 1 through 5 shared library; these load modules are not supported.</td>
</tr>
<tr>
<td>Load modules from OS PL/I Version 1 Release 5.1 and the Version 2 shared library are supported; however, you must rebuild the shared library once under Language Environment.</td>
<td></td>
</tr>
</tbody>
</table>

Migrating assembler programs to Language Environment

To run assembler programs with Language Environment, you must ensure the assembler programs adhere to conventions for items such as register and storage usage, condition handling, and accessing input parameters. For example, assembler programs must set a valid 31 bit address in the save area back chain.

Language Environment provides several assembler macros, which your assembler programs should use to perform tasks such as entering and exiting assembler routines and mapping Language Environment data areas. For example, when you use the CEEENTRY and CEETERM macros, Language Environment automatically initializes and terminates, respectively, the execution environment for the application. In addition, when the Language Environment environment is established for the main assembler program, that environment is also established for any other routines that may be called later.


For more information about assembler considerations when assembler programs are used with COBOL, see the appropriate version of the COBOL migration guide in the [Enterprise COBOL for z/OS library](http://www-01.ibm.com/support/docview.wss?uid=swg27036733).
Chapter 4. Choosing runtime options for compatible behavior

This topic provides information on how Language Environment runtime options differ from runtime options that are specific to a high-level language (HLL). For more information about runtime options, see the following publications:

- z/OS Language Environment Customization
- z/OS Language Environment Programming Reference

Differences between runtime options

Language Environment provides a set of runtime options for applications. These options are processed at the enclave level and allow you to control many aspects of the Language Environment environment. The options comparison tables show how Language Environment runtime options differ from the runtime options that are specific to C, COBOL, Fortran, and PL/I (if a HLL runtime option is not listed in a table, you can assume it operates under Language Environment in the same way it did before Language Environment):

<table>
<thead>
<tr>
<th>High-level language</th>
<th>Language Environment option information</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>Table 6 on page 18</td>
</tr>
<tr>
<td>COBOL</td>
<td>Table 7 on page 18</td>
</tr>
<tr>
<td>Fortran</td>
<td>Table 8 on page 20</td>
</tr>
<tr>
<td>PL/I</td>
<td>Table 9 on page 21</td>
</tr>
</tbody>
</table>

C and Language Environment runtime options comparison

Table 6. C and Language Environment runtime options

<table>
<thead>
<tr>
<th>C option</th>
<th>Language Environment equivalent</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISAINC</td>
<td>STACK</td>
<td>If you do not change the C/370 runtime option ISAINC, you will receive a warning message during execution.</td>
</tr>
<tr>
<td>ISASIZE</td>
<td>STACK</td>
<td>If you do not change the C/370 runtime option ISASIZE, you will receive a warning message during execution.</td>
</tr>
<tr>
<td>LANGUAGE</td>
<td>NATLANG</td>
<td>Mixed-case and uppercase US English and Japanese are supported. If you do not change the C/370 runtime option LANGUAGE, you will receive a warning message during execution.</td>
</tr>
<tr>
<td>REPORT</td>
<td>NOREPORT</td>
<td>RPTSTG(ON</td>
</tr>
<tr>
<td>SPIE</td>
<td>NOSPIE</td>
<td>TRAP(ON,SPIE)</td>
</tr>
<tr>
<td>STAE</td>
<td>NOSTAE</td>
<td>TRAP(OFF)</td>
</tr>
</tbody>
</table>
Choosing compatible runtime options

**COBOL and Language Environment runtime options comparison**

<table>
<thead>
<tr>
<th>COBOL option</th>
<th>Language Environment equivalent</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIXBLD</td>
<td>NOAIXBLD</td>
<td>AIXBLD</td>
</tr>
<tr>
<td>DEBUG</td>
<td>NODEBUG</td>
<td>DEBUG</td>
</tr>
<tr>
<td>FLOW</td>
<td>NOFLOW</td>
<td>FLOW</td>
</tr>
<tr>
<td>LANGUAGE</td>
<td>NATLANG</td>
<td>NATLANG replaces LANGUAGE, which is a VS COBOL II installation option. You can select a national language at runtime or installation time by using the NATLANG option.</td>
</tr>
<tr>
<td>LIBKEEP</td>
<td>NOLIBKEEP</td>
<td>Not applicable</td>
</tr>
<tr>
<td>MIXRES</td>
<td>NOMIXRES</td>
<td>Not applicable</td>
</tr>
<tr>
<td>QUEUE</td>
<td>Not applicable</td>
<td>QUEUE is not supported under Language Environment.</td>
</tr>
</tbody>
</table>
### Choosing compatible runtime options

Table 7. COBOL and Language Environment runtime options (continued)

<table>
<thead>
<tr>
<th>COBOL option</th>
<th>Language Environment equivalent</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTEREUS</td>
<td>NORTEREUS</td>
<td>RTEREUS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIMVRD</td>
<td>NOSIMVRD</td>
<td>SIMVRD</td>
</tr>
<tr>
<td>SPOUT</td>
<td>NOSPOUT</td>
<td>RPTOPTS(ON</td>
</tr>
<tr>
<td>SSRANGE</td>
<td>NOSSRANGE</td>
<td>CHECK(ON</td>
</tr>
<tr>
<td>STAE</td>
<td>NOSTAE</td>
<td>TRAP(ON,SPIE) TRAP(OFF)</td>
</tr>
<tr>
<td>UPSI</td>
<td>UPSI</td>
<td>UPSI provides behavior compatible with the VS COBOL II UPSI option.</td>
</tr>
<tr>
<td>WSCLEAR</td>
<td>NOWSCLEAR</td>
<td>STORAGE(00,..)</td>
</tr>
</tbody>
</table>
Choosing compatible runtime options

Fortran and Language Environment runtime options comparison

Table 8. Fortran and Language Environment runtime options

<table>
<thead>
<tr>
<th>Fortran option</th>
<th>Language Environment equivalent</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSDUMP</td>
<td>NOABSDUMP</td>
<td>TERMTHDACT</td>
</tr>
<tr>
<td>AUTOTASK</td>
<td>NOAUTOTASK</td>
<td>AUTOTASK</td>
</tr>
<tr>
<td>CNVIOERR</td>
<td>NOCNVIOERR</td>
<td>Not applicable</td>
</tr>
<tr>
<td>DEBUG</td>
<td>NODEBUG</td>
<td>Not applicable</td>
</tr>
<tr>
<td>DEBUNIT</td>
<td>Not applicable</td>
<td>There is no Language Environment equivalent for DEBUNIT.</td>
</tr>
<tr>
<td>ECPACK</td>
<td>NOECPACK</td>
<td>Not applicable</td>
</tr>
<tr>
<td>ERRUNIT</td>
<td>ERRUNIT</td>
<td>ERRUNIT provides behavior compatible with VS FORTRAN Version 2.</td>
</tr>
<tr>
<td>FAIL</td>
<td>ABTERMENC</td>
<td>ABTERMENC replaces FAIL, but there is no automatic mapping. ABTERMENC controls whether a condition of severity 2 or greater is terminated with a return code or an abend. ABTERMENC(RETCODE) is similar to FAIL(RC), and ABTERMENC(ABEND) is similar to FAIL(ABEND).</td>
</tr>
<tr>
<td>FILEHIST</td>
<td>NOFILEHIST</td>
<td>FILEHIST</td>
</tr>
<tr>
<td>INQPCOPN</td>
<td>NOINQPCOPN</td>
<td>INQPCOPN</td>
</tr>
<tr>
<td>JOINIT</td>
<td>NOJOINIT</td>
<td>Not applicable</td>
</tr>
</tbody>
</table>
### Fortran and Language Environment runtime options (continued)

<table>
<thead>
<tr>
<th>Fortran option</th>
<th>Language Environment equivalent</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCSTATUS / NOOCSTATUS</td>
<td>OCSTATUS / NOOCSTATUS</td>
<td>OCSTATUS / NOOCSTATUS provides behavior compatible with VS FORTRAN Version 2.</td>
</tr>
<tr>
<td>PARALLEL / NOPARALLEL</td>
<td>Not applicable</td>
<td>There is no Language Environment equivalent for PARALLEL / NOPARALLEL. Parallel programs cannot be run with Language Environment.</td>
</tr>
<tr>
<td>PC / NOPC</td>
<td>PC / NOPC</td>
<td>PC specifies that Fortran static common blocks with the same name but in different load modules do not refer to the same storage.</td>
</tr>
<tr>
<td>PRTUNIT</td>
<td>PRTUNIT</td>
<td>PRTUNIT provides behavior compatible with VS FORTRAN Version 2.</td>
</tr>
<tr>
<td>PTRACE / NOPTRACE</td>
<td>Not applicable</td>
<td>There is no Language Environment equivalent for PTRACE / NOPTRACE. Parallel programs cannot be run with Language Environment.</td>
</tr>
<tr>
<td>PUNUNIT</td>
<td>PUNUNIT</td>
<td>PUNUNIT provides behavior compatible with VS FORTRAN Version 2.</td>
</tr>
<tr>
<td>RDRUNIT</td>
<td>RDRUNIT</td>
<td>RDRUNIT provides behavior compatible with VS FORTRAN Version 2.</td>
</tr>
<tr>
<td>RECPAD / NORECPAD</td>
<td>RECPAD(OFF</td>
<td>NONE</td>
</tr>
<tr>
<td>SPIE / NOSPIE STAE / NOSTAE</td>
<td>TRAP(ON,SPIE) TRAP(OFF)</td>
<td>If either SPIE or STAE is specified or defaulted in input, TRAP is set to TRAP(ON,SPIE). If both NOSPIE and NOSTAE are specified, TRAP is set to TRAP(OFF). TRAP(ON,SPIE) is the recommended setting.</td>
</tr>
<tr>
<td>XUFLOW / NOXUFLOW</td>
<td>XUFLOW(ON</td>
<td>AUTO) XUFLOW(OFF)</td>
</tr>
</tbody>
</table>

### PL/I and Language Environment runtime options comparison

<table>
<thead>
<tr>
<th>PL/I option</th>
<th>Language Environment equivalent</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>COUNT / NOCOUNT</td>
<td>Not applicable</td>
<td>There is no Language Environment equivalent for COUNT / NOCOUNT. It is not processed but produces an informational message.</td>
</tr>
</tbody>
</table>
### Choosing compatible runtime options

<table>
<thead>
<tr>
<th>PL/I option</th>
<th>Language Environment equivalent</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLOW</td>
<td>NOFLOW</td>
<td>Not applicable</td>
</tr>
<tr>
<td>ISAINC</td>
<td></td>
<td>STACK, THREADSTACK, or PLITASKCOUNT</td>
</tr>
<tr>
<td>ISASIZE</td>
<td></td>
<td>STACK, THREADSTACK, or PLITASKCOUNT</td>
</tr>
<tr>
<td>LANGUAGE</td>
<td></td>
<td>NATLANG</td>
</tr>
<tr>
<td>REPORT</td>
<td>NOREPORT</td>
<td>RPTSTG(ON</td>
</tr>
<tr>
<td>SPIE</td>
<td>NOSPIE</td>
<td>TRAP(ON</td>
</tr>
<tr>
<td>STAE</td>
<td>NOSTAE</td>
<td>TRAP(ON</td>
</tr>
<tr>
<td>TASKHEAP</td>
<td></td>
<td>THREADHEAP</td>
</tr>
</tbody>
</table>
Chapter 5. Other HLL migration considerations

This topic includes migration concerns that are language-specific, such as: differences in how Language Environment and an HLL handle return codes, runtime messages, entry files, and user exits. For more information about these considerations, see the following references:

- z/OS XL C/C++ Compiler and Runtime Migration Guide for the Application Programmer
- The appropriate version of the COBOL migration guide in the Enterprise COBOL for z/OS library (http://www-01.ibm.com/support/docview.wss?uid=swg27036733)
- The IBM Enterprise PL/I for z/OS library (http://www.ibm.com/support/docview.wss?uid=swg27036735)
- z/OS Language Environment Programming Guide

C considerations

The following topics list some sample migration problems. For a complete list of migration considerations, see one of the C migration guides listed in the preceding topic.

Standard streams

Under z/OS Language Environment there is no longer an automatic association of ddnames SYSTERM, SYSERR, SYSPRINT with stderr. Command line redirection of the type 1>&2 is necessary in batch to cause stderr and stdout to share a device.

In C/370 Version 1 and Version 2, you could override the destination of error messages by redirecting stderr. Language Environment determines the destination of all messages from the new MSGFILE runtime option. For more information about the MSGFILE option, see z/OS Language Environment Programming Guide.

Passing command line parameters

In C/370 Version 1 or Version 2, if an error was detected with the parameters being passed to the main program, the program terminated with a return code of 8 and a message indicating the reason the program terminated. For example, if there was an error in the redirection parameters, the message would indicate that the program had terminated because of a redirection error. Under Language Environment, the same message is displayed, but the program also terminates with a 4093 abend, reason code 52 (X'34'). For more information about reason codes, see z/OS Language Environment Debugging Guide.

User exits

If CEEBXITA and IBMBXITA are present in a relinked C/370 Version 1 or Version 2 module, CEEBXITA will take precedence over IBMBXITA.

Time functions

If you are migrating from IBM C/370 (Version 1 or Version 2) or AD/Cycle C/370 (Version 1 Release 1 or Version 1 Release 2), you should be aware of the following change in time functions.
Other considerations

- The `ctime()`, `localtime()`, and `mktime()` functions will return Coordinated Universal Time (UTC) time unless customized locale information is available. When you customize the locale, time functions preserve the time and date and correctly adjust for daylight time on a given date. See [z/OS XL C/C++ Programming Guide](https://www.ibm.com/support/docview.wss?uid=swg27044344) for more information about environment variables and customizing locale information.

- In POSIX and non-POSIX applications, you can use the `TZ` environment variable to supply the necessary time zone information for your location. Previously, for non-POSIX applications, you could supply customized locale information only by setting time zone and daylight information in the _LC_T00_ locale category.

Load modules that invoke a Debugging Tool

C/370 library application load modules that use `ctest()` to invoke the Debug Tool must be relinked to run with Language Environment. The old library object, `@@CTEST`, must be replaced, as described in [z/OS XL C/C++ Compiler and Runtime Migration Guide for the Application Programmer](https://www.ibm.com/support/docview.wss?uid=swg22403682) and [IBM C for VM/ESA Compiler and Run-Time Migration Guide](https://www.ibm.com/support/docview.wss?uid=swg27005254). After you replace the old objects, the new modules will run with Language Environment.

Prefix of perror() and strerror() messages in C

With Language Environment, all `perror()` and `strerror()` messages in C contain a prefix. With C/370 Version 1 and Version 2 there was no prefix on these messages. The prefix is EDCxxxxa, where xxxx is a number (always 5xxx) and the a is I, W, or E. See [z/OS Language Environment Runtime Messages](https://www.ibm.com/support/docview.wss?uid=swg21282105) for a list of messages.

AMODE errors from ILCs

In ILC applications of C/370 Version 1 or Version 2 and VS COBOL II, if C/370 was running at AMODE 31 and COBOL was running at AMODE 24, an error was produced (2052) and the application failed. Under Language Environment, the call will fail but the message will be EDC5052, protection exception.

PL/I considerations

The following topics describe some of the items you should consider when migrating PL/I applications to Language Environment.

Dumps


Condition handling

In general, PL/I condition handling functions in the same way when running under Language Environment. However, the issuing of diagnostic messages may vary. For example, the diagnostic message for an ERROR condition is issued only if there is no ERROR ON-unit established, or if the ERROR ON-unit does not recover from the condition by using a GOTO out of block. However, for other PL/I conditions whose implicit action includes printing a message and raising the ERROR condition, the message is issued before control is given to an established ERROR ON-unit.
Other considerations

User exits

The OS PL/I Version 2 assembler user exits IBMBXITA and IBMFXITA are supported by PL/I for MVS & VM for compatibility. However, the Language Environment user exit CEEBINT should be used instead. Only CEEBINT is supported by VisualAge PL/I for OS/390.

Also, the OS PL/I Version 2 high-level language user exit IBMBINT is not recommended; it is supported only for compatibility. Use the Language Environment high-level language user exit, CEEBINT, instead. Only CEEBINT is supported by VisualAge PL/I for OS/390. See VisualAge PL/I for OS/390 Compiler and Run-Time Migration Guide, Enterprise PL/I for z/OS, V3R9, Migration Guide, or PL/I for MVS & VM Compiler and Run-Time Migration Guide for detailed information. See z/OS Language Environment Programming Guide for more information about the Language Environment user exits.

SYSPRINT

In PL/I, runtime messages are directed, by default, to the Language Environment MSGFILE rather than to SYSPRINT. Runtime user output is still directed to SYSPRINT. If you want runtime messages to go to SYSPRINT, specify the MSGFILE(SYSPRINT) runtime option. In this case, SYSPRINT can contain both user output and runtime output. For more information about the MSGFILE runtime option, see z/OS Language Environment Programming Reference.

If you specify a RECSIZE value that is not consistent with the LRECL of the data set or with the LRECL on the DD statement, the PL/I runtime library will diagnose this with an UNDEFINEDFILE condition with ONCODE=81. You must change the JCL to ensure that the values are the same or remove the LRECL value from the DD statement.

For DB2 UDB for z/OS Version 8 and DB2 Version 9.1 for z/OS customers, job steps that execute program DSNTEP2 or DSNTEP4 will experience user abend 4038. The user abend 4038 happens because UNDEFINEDFILE condition with ONCODE=81 error when the SYSPRINT DD specifies an LRECL that does not match the RECSIZE specified by DSNTEP2 and DSNTEP4 in the PAGEWIDTH constant. PAGEWIDTH is typically 133 but can be changed in the source code for DSNTEP2 and DSNTEP4. If you experience the abend and do not know the PAGEWIDTH setting, remove the LRECL from the SYSPRINT DD in job steps that execute DSNTEP2 or DSNTEP4.

Format and content of messages

The format and content of runtime messages is different for PL/I applications that run with Language Environment. Differences include the following items:

- The message number in the message prefix is now four digits instead of three digits.
- The message severity in the message prefix can now be C, E, I, S, or W.
- The message text of some mixed-case English and Japanese messages has been enhanced.

You must modify your applications if they analyze the runtime output. See z/OS Language Environment Programming Reference for more information about using and handling messages.
General considerations

This topic describes other items you should consider when migrating a pre-Language Environment HLL application to an application that conforms to Language Environment.

Return and reason codes

Some return and reason codes will differ when running under Language Environment. JCL and EXECs that are affected by them must be changed accordingly. See z/OS Language Environment Debugging Guide information for details about return and reason codes.

Storage reports

The output of the runtime storage report is different when running with Language Environment. For information about the RPTSTG runtime option, see z/OS Language Environment Programming Reference. For an example of the storage report, see z/OS Language Environment Debugging Guide.

Stream I/O

If you choose the LINESIZE option, and you provide a record size value that is too small to hold the LINESIZE (taking into account the record format and appropriate control byte overhead), the UNDEFINEDFILE condition is raised.

This behavior is different from previous Language Environment releases. In previous releases, the following was true:

For DD SYSOUT= files except SYSPRINT, if you chose the LINESIZE option, and you provided a record size value that is too small to hold the LINESIZE (taking into account the record format and appropriate control byte overhead), the LINESIZE is used to determine a new record size that matched the given LINESIZE. For DD DSN= files, the UNDEFINEDFILE condition is raised.
Appendix. Accessibility

Accessible publications for this product are offered through the z/OS Information Center, which is available at www.ibm.com/systems/z/os/zos/bkserv/.

If you experience difficulty with the accessibility of any z/OS information, please send a detailed message to mhvrcfs@us.ibm.com or to the following mailing address:

IBM Corporation
Attention: MHVRCFS Reader Comments
Department H6MA, Building 707
2455 South Road
Poughkeepsie, NY 12601-5400
USA

Accessibility features

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. The major accessibility features in z/OS enable users to:

- Use assistive technologies such as screen readers and screen magnifier software
- Operate specific or equivalent features using only the keyboard
- Customize display attributes such as color, contrast, and font size.

Using assistive technologies

Assistive technology products, such as screen readers, function with the user interfaces found in z/OS. Consult the assistive technology documentation for specific information when using such products to access z/OS interfaces.

Keyboard navigation of the user interface

Users can access z/OS user interfaces using TSO/E or ISPF. Refer to z/OS TSO/E Primer, z/OS TSO/E User's Guide, and z/OS ISPF User's Guide Vol I for information about accessing TSO/E and ISPF interfaces. These guides describe how to use TSO/E and ISPF, including the use of keyboard shortcuts or function keys (PF keys). Each guide includes the default settings for the PF keys and explains how to modify their functions.

Dotted decimal syntax diagrams

Syntax diagrams are provided in dotted decimal format for users accessing the z/OS Information Center using a screen reader. In dotted decimal format, each syntax element is written on a separate line. If two or more syntax elements are always present together (or always absent together), they can appear on the same line, because they can be considered as a single compound syntax element.

Each line starts with a dotted decimal number; for example, 3 or 3.1 or 3.1.1. To hear these numbers correctly, make sure that your screen reader is set to read out punctuation. All the syntax elements that have the same dotted decimal number (for example, all the syntax elements that have the number 3.1) are mutually
exclusive alternatives. If you hear the lines 3.1 USERID and 3.1 SYSTEMID, you
know that your syntax can include either USERID or SYSTEMID, but not both.

The dotted decimal numbering level denotes the level of nesting. For example, if a
syntax element with dotted decimal number 3 is followed by a series of syntax
elements with dotted decimal number 3.1, all the syntax elements numbered 3.1
are subordinate to the syntax element numbered 3.

Certain words and symbols are used next to the dotted decimal numbers to add
information about the syntax elements. Occasionally, these words and symbols
might occur at the beginning of the element itself. For ease of identification, if the
word or symbol is a part of the syntax element, it is preceded by the backslash (\)
character. The * symbol can be used next to a dotted decimal number to indicate
that the syntax element repeats. For example, syntax element *FILE with dotted
decimal number 3 is given the format 3 * FILE. Format 3* FILE indicates that
syntax element FILE repeats. Format 3* * FILE indicates that syntax element *
FILE repeats.

Characters such as commas, which are used to separate a string of syntax
elements, are shown in the syntax just before the items they separate. These
characters can appear on the same line as each item, or on a separate line with the
same dotted decimal number as the relevant items. The line can also show another
symbol giving information about the syntax elements. For example, the lines 5.1*,
5.1 LASTRUN, and 5.1 DELETE mean that if you use more than one of the
LASTRUN and DELETE syntax elements, the elements must be separated by a
comma. If no separator is given, assume that you use a blank to separate each
syntax element.

If a syntax element is preceded by the % symbol, this indicates a reference that is
defined elsewhere. The string following the % symbol is the name of a syntax
fragment rather than a literal. For example, the line 2.1 %OP1 means that you
should refer to separate syntax fragment OP1.

The following words and symbols are used next to the dotted decimal numbers:
• ? means an optional syntax element. A dotted decimal number followed by the ?
symbol indicates that all the syntax elements with a corresponding dotted
decimal number, and any subordinate syntax elements, are optional. If there is
only one syntax element with a dotted decimal number, the ? symbol is
displayed on the same line as the syntax element, (for example 5? NOTIFY). If
there is more than one syntax element with a dotted decimal number, the ?
symbol is displayed on a line by itself, followed by the syntax elements that are
optional. For example, if you hear the lines 5 ?, 5 NOTIFY, and 5 UPDATE, you
know that syntax elements NOTIFY and UPDATE are optional; that is, you can
choose one or none of them. The ? symbol is equivalent to a bypass line in a
railroad diagram.
• ! means a default syntax element. A dotted decimal number followed by the !
symbol and a syntax element indicates that the syntax element is the default
option for all syntax elements that share the same dotted decimal number. Only
one of the syntax elements that share the same dotted decimal number can
specify a ! symbol. For example, if you hear the lines 2? FILE, 2.1! (KEEP), and
2.1 (DELETE), you know that (KEEP) is the default option for the FILE keyword.
In this example, if you include the FILE keyword but do not specify an option,
default option KEEP will be applied. A default option also applies to the next
higher dotted decimal number. In this example, if the FILE keyword is omitted,
default FILE(KEEP) is used. However, if you hear the lines 2? FILE, 2.1, 2.1.1!
(KEEP), and 2.1.1 (DELETE), the default option KEEP only applies to the next higher dotted decimal number, 2.1 (which does not have an associated keyword), and does not apply to 2? FILE. Nothing is used if the keyword FILE is omitted.

- * means a syntax element that can be repeated 0 or more times. A dotted decimal number followed by the * symbol indicates that this syntax element can be used zero or more times; that is, it is optional and can be repeated. For example, if you hear the line 5.1* data area, you know that you can include one data area, more than one data area, or no data area. If you hear the lines 3*, 3 HOST, and 3 STATE, you know that you can include HOST, STATE, both together, or nothing.

**Note:**

1. If a dotted decimal number has an asterisk (*) next to it and there is only one item with that dotted decimal number, you can repeat that same item more than once.

2. If a dotted decimal number has an asterisk next to it and several items have that dotted decimal number, you can use more than one item from the list, but you cannot use the items more than once each. In the previous example, you could write HOST STATE, but you could not write HOST HOST.

3. The * symbol is equivalent to a loop-back line in a railroad syntax diagram.

- + means a syntax element that must be included one or more times. A dotted decimal number followed by the + symbol indicates that this syntax element must be included one or more times; that is, it must be included at least once and can be repeated. For example, if you hear the line 6.1+ data area, you must include at least one data area. If you hear the lines 2+, 2 HOST, and 2 STATE, you know that you must include HOST, STATE, or both. Similar to the * symbol, the + symbol can only repeat a particular item if it is the only item with that dotted decimal number. The + symbol, like the * symbol, is equivalent to a loop-back line in a railroad syntax diagram.
This information was developed for products and services offered in the U.S.A. or elsewhere.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user’s responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.
IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

Site Counsel
IBM Corporation
2455 South Road
Poughkeepsie, NY 12601-5400
USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of 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.

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

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

COPYRIGHT LICENSE:

This information might contain sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Policy for unsupported hardware

Various z/OS elements, such as DFSMS, HCD, JES2, JES3, and MVS, contain code that supports specific hardware servers or devices. In some cases, this device-related element support remains in the product even after the hardware devices pass their announced End of Service date. z/OS may continue to service element code; however, it will not provide service related to unsupported hardware devices. Software problems related to these devices will not be accepted
for service, and current service activity will cease if a problem is determined to be associated with out-of-support devices. In such cases, fixes will not be issued.

Minimum supported hardware

The minimum supported hardware for z/OS releases identified in z/OS announcements can subsequently change when service for particular servers or devices is withdrawn. Likewise, the levels of other software products supported on a particular release of z/OS are subject to the service support lifecycle of those products. Therefore, z/OS and its product publications (for example, panels, samples, messages, and product documentation) can include references to hardware and software that is no longer supported.

- For information about software support lifecycle, see: [IBM Lifecycle Support for z/OS](http://www.ibm.com/software/support/systemsz/lifecycle/)
- For information about currently-supported IBM hardware, contact your IBM representative.

Programming Interface information

This book documents intended Programming Interfaces that allow the customer to write programs to obtain the services of Language Environment in z/OS.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).
## Index

### A
- accessibility 27
- contact IBM 27
- features 27
  - assembler programs
    - migrating to Language Environment 15
  - assistive technologies 27

### C
- C routines
  - migrating to Language Environment 13
- CEEPRMxx parmlib member 6
- CLER runtime options report 8
- COBOL programs
  - migrating to Language Environment 13
  - compatibility
    - with previous runtime libraries 11
    - conversion table removal 7

### D
- D CEE 7
- dumps (PLIDUMP) 24

### F
- Fortran routines
  - migrating to Language Environment 13

### H
- HEAPCHK runtime option
  - changes to 9

### I
- ILC applications
  - migrating to Language Environment 12

### K
- keyboard
  - navigation 27
  - PF keys 27
  - shortcut keys 27

### L
- Language Environment (continued)
  - planning to migrate to 1
- Language Environment
  - migrating
    - assembler programs to Language Environment 15
    - C routines to Language Environment 13
    - COBOL programs to Language Environment 13
    - Fortran routines to Language Environment 13
    - ILC applications to Language Environment 12
    - PL/I routines to Language Environment 14
  - migration
    - checklist for 1
    - from another Language Environment release 5
    - from other runtime environments 11
    - migration actions
      - for each release 5

### N
- navigation
  - keyboard 27
- Notices 31

### P
- PL/I routines
  - migrating to Language Environment 14

### R
- runtime options
  - choosing for compatible behavior 17
  - comparison between C and Language Environment 17
  - comparison between COBOL and Language Environment 18
  - comparison between Fortran and Language Environment 20
  - comparison between PL/I and Language Environment 21
  - differences between 17
  - runtime options report 7

### S
- sending comments to IBM xi
- shortcut keys 27
- Summary of changes xiii
- Trademarks 33
- Unicode Services 7
- user exits 25
- user interface
  - ISPF 27
  - TSO/E 27