Enterprise PL/I for z/OS
IBM Developer for z Systems PL/I for Windows

Messages and Codes

Version 5 Release 2 Modification 4

GC27-8950-01

Enterprise PL/I for z/OS
IBM Developer for z Systems PL/I for Windows

Messages and Codes

Version 5 Release 2 Modification 4

GC27-8950-01

Note
Before using this information and the product it supports, be sure to read the general information under "Notices" on page 165.

Second Edition (December 2018)

This edition applies to Version 5 Release 2 of Enterprise PL/I for z/OS (5655-PL5), and Version 9.1 of IBM Developer for z Systems (formerly IBM Rational Developer for System z) PL/I for Windows, and to any subsequent releases until otherwise indicated in new editions or technical newsletters. Make sure you are using the correct edition for the level of the product.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address below.

A form for readers' comments is provided at the back of this publication. If the form has been removed, address your comments to:

IBM Corporation, Department H150/090
555 Bailey Ave.
San Jose, CA, 95141-1099
United States of America

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

© Copyright IBM Corporation 1999, 2018.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

About this book v
Compiler and preprocessor messages v
How to send your comments vi
Accessibility vii

Chapter 1. Compiler Informational Messages (1000-1076, 2800-2999). . . . 1

Chapter 2. Compiler Warning Messages (1078-1225, 2600-2799) 7

Chapter 3. Compiler Error Messages (1226-1499, 2400-2599) 27

Chapter 4. Compiler Severe Messages (1500-2399) 53

Chapter 5. MACRO, CICS, and SQL Preprocessor Messages (3000-3999) . 111

Chapter 6. Code Generation Messages (5000-5999) 137

Chapter 7. Condition codes 141
Condition codes 1 through 500 141
Condition codes 501 through 1000 146
Condition codes 1001 through 1499 149
Condition codes 1500 through 2000 151
Condition codes 2001 through 2500 158
Condition codes 3000 through 4000 160
Condition codes 4001 through 9999 161

Notices 165
Trademarks 166

Bibliography 167
PL/I publications 167
Related publications 167

© Copyright IBM Corp. 1999, 2018

About this book

This book is for PL/I programmers and system programmers. It helps you understand compiler and preprocessor messages.

Compiler and preprocessor messages

This guide lists the compiler messages in numerical order. These messages are also listed in numerical order in the output following the source program and in any other listings produced by the compiler.

Format of messages

In your compilation output, each compiler message, with the exception of the code generation messages in the range 5000-5999, starts with `IBMnnnnI X` where:

- `IBM` indicates that the message is a PL/I message.
- `nnnn` is the number of the message.
- The closing letter `I` indicates that no system operator action is required.
- The `X` represents a severity code.

In some catastrophic situations, such as not being able to open SYSPRINT, the compiler might not follow the last two of the preceding rules.

In this guide, messages are listed numerically. Each compiler message in this section has the form `IBMnnnnI X` where `X` is the severity code.

Severity codes can be any of the following: I, W, E, S, or U.

These severity codes indicate the following. (Note that the return codes listed are the highest return code generated.)

I An ***informational*** message (RC=0) indicates that the compiled program should run correctly. The compiler might inform you of a possible inefficiency in your code or some other condition of interest.

W A ***warning*** message (RC=4) warns you that a statement might be in error (warning) even though it is syntactically valid. The compiled program should run correctly, but might produce different results than expected or be significantly inefficient.

E An ***error*** message (RC=8) describes a simple error fixed by the compiler. The compiled program should run correctly, but might produce different results than expected.

S A ***severe*** error message (RC=12) describes an error not fixed by the compiler. If the program is compiled and an object module is produced, it should not be used.

U An ***unrecoverable*** error message (RC=16) signifies an error that forces termination of the compilation. An object module is not successfully created.

Compiler messages are printed in groups according to these severity levels and to the component that produced them.

© Copyright IBM Corp. 1999, 2018

The code generation messages (those in the range 5000-5999) start with `IBMnnnn` where:

- `IBM` indicates that the message is a PL/I message.
- `nnnn` is the number of the message.

Under batch, the code generation messages are written to the STDOUT DD data set, while all other messages appear in the listing which is written to the SYSPRINT DD data set. Under z/OS UNIX, the code generation messages are written to stdout, while all other messages appear in the listing and are also written to stdout.

The compiler FLAG option suppresses the listing of messages in the compiler listing. You can find a description of the FLAG option in *Enterprise PL/I for z/OS Programming Guide*.

Message inserts

Many of the compiler messages contain message inserts indicating where the compiler inserts information when it prints the message. These inserts are emphasized in the messages in this section using *italics*.

Contacting IBM for support

If you contact IBM for programming support for a compiler error, it is useful to have a listing of your source program available. To make the analysis of any potential problem easier, it is best if that listing is created with the options: INSOURCE MACRO OPTIONS SOURCE.

How to send your comments

Your feedback is important in helping us to provide accurate, high-quality information. If you have comments about this document or any other PL/I documentation, contact us in one of these ways:

- Use the Online Readers' Comment Form at

  `www.ibm.com/software/awdtools/rcf/`

  or send an e-mail to

  `compinfo@cn.ibm.com`

  Be sure to include the name of the document, the publication number of the document, the version of PL/I, and, if applicable, the specific location (for example, page number) of the text that you are commenting on.
- Fill out the Readers' Comment Form at the back of this document, and return it by mail or give it to an IBM representative. If the form has been removed, address your comments to:

  International Business Machines Corporation
  Reader Comments
  H150/090
  555 Bailey Avenue
  San Jose, CA 95141-1003
  USA
- Fax your comments to this U.S. number: (800)426-7773.

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

Accessibility

Accessibility features assist users who have a disability, such as restricted mobility or limited vision, to use information technology content successfully. The accessibility features in z/OS® provide accessibility for Enterprise PL/I.

Accessibility features

z/OS includes the following major accessibility features:

- Interfaces that are commonly used by screen readers and screen-magnifier software
- Keyboard-only navigation
- Ability to customize display attributes such as color, contrast, and font size

z/OS uses the latest W3C Standard, WAI-ARIA 1.0 (http://www.w3.org/TR/wai-aria/), to ensure compliance to US Section 508 (http://www.access-board.gov/guidelines-and-standards/communications-and-it/about-the-section-508-standards/section-508-standards) and Web Content Accessibility Guidelines (WCAG) 2.0 (http://www.w3.org/TR/WCAG20/). To take advantage of accessibility features, use the latest release of your screen reader in combination with the latest web browser that is supported by this product.

The Enterprise PL/I online product documentation in IBM® Knowledge Center is enabled for accessibility. The accessibility features of IBM Knowledge Center are described at http://www.ibm.com/support/knowledgecenter/en/about/releasenotes.html.

Keyboard navigation

Users can access z/OS user interfaces by using TSO/E or ISPF.

Users can also access z/OS services by using IBM Developer for z Systems.

For information about accessing these interfaces, see the following publications:

- *z/OS TSO/E Primer* (http://publib.boulder.ibm.com/cgi-bin/bookmgr/BOOKS/ikj4p120)
- *z/OS TSO/E User's Guide* (http://publib.boulder.ibm.com/cgi-bin/bookmgr/BOOKS/ikj4c240/APPENDIX1.3)
- *z/OS ISPF User's Guide Volume I* (http://publib.boulder.ibm.com/cgi-bin/bookmgr/BOOKS/ispzug70)
- IBM Rational® Developer for System z® Knowledge Center (http://www.ibm.com/support/knowledgecenter/SSQ2R2/rdz_welcome.html?lang=en)

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.

Interface information

The Enterprise PL/I online product documentation is available in IBM Knowledge Center, which is viewable from a standard web browser.

PDF files have limited accessibility support. With PDF documentation, you can use optional font enlargement, high-contrast display settings, and can navigate by keyboard alone.

To enable your screen reader to accurately read syntax diagrams, source code examples, and text that contains the period or comma PICTURE symbols, you must set the screen reader to speak all punctuation.

Assistive technology products work with the user interfaces that are found in z/OS. For specific guidance information, see the documentation for the assistive technology product that you use to access z/OS interfaces.

Related accessibility information

In addition to standard IBM help desk and support websites, IBM has established a TTY telephone service for use by deaf or hard of hearing customers to access sales and support services:

TTY service
800-IBM-3383 (800-426-3383)
(within North America)

IBM and accessibility

For more information about the commitment that IBM has to accessibility, see IBM Accessibility (www.ibm.com/able).

Chapter 1. Compiler Informational Messages (1000-1076, 2800-2999)

IBM1018I I *option-name* **should be specified within OPTIONS, but is accepted as is.**

Explanation: This message is used in building the options listing.

IBM1035I I **The next statement was merged with this statement.**

Explanation: The statement following the statement for which this message was issued were merged with that statement.

IBM1036I I **The next** *statement-count* **statements were merged with this statement.**

Explanation: The specified number of statements following the statement for which this message was issued were merged with that statement.

IBM1038I I *note*

Explanation: This message is used to report back end informational messages.

IBM1039I I **Variable** *variable name* **is implicitly declared.**

Explanation: All variables should be declared except for contextual declarations of built-in functions, SYSPRINT and SYSIN.

IBM1040I I *note*

Explanation: This message is used by %NOTE statements with a return code of 0.

IBM1041I I **Comment spans** *line-count* **lines.**

Explanation: A comment ends on a different line than it begins. This may indicate that an end-of-comment delimiter is missing.

IBM1042I I **String spans** *line-count* **lines.**

Explanation: A string ends on a different line than it begins. This may indicate that a closing quote is missing.

IBM1043I I *variable name* **is contextually declared as** *attribute***.**

Explanation: There is no declare statement for the named variable, but it has been given the indicated attribute because of its usage. For instance, if the variable is used as a locator, it will be given the POINTER attribute.

IBM1044I I **FIXED BINARY with precision 7 or less is mapped to 1 byte.**

Explanation: The OS/370 PL/I and PL/I for MVS compilers would have mapped this to 2 bytes.

IBM1046I I **UNSPEC applied to an array is handled as a scalar reference.**

Explanation: The OS/370 PL/I and PL/I for MVS compilers would have handled UNSPEC applied to an array as an array of scalars.

IBM1047I I **ORDER option may inhibit optimization.**

Explanation: If the ORDER option applies to a block, optimization is likely to be inhibited, especially if the block contains ON-units that refer to variables declared outside the ON-unit.

IBM1048I I **GET/PUT DATA without a data-list inhibits optimization.**

Explanation: A GET DATA statement can alter almost any variable, and a PUT DATA statement requires almost all variables to be stored home anytime a PUT DATA statement might be executed. Both of these requirements inhibit optimization.

IBM1050I I **INITIAL attribute for RESERVED STATIC is ignored.**

Explanation: The INITIAL attribute has been specified for a variable with the attributes RESERVED STATIC. Unless such a variable is listed in the EXPORTS clause of a PACKAGE statement, the variable will not be initialized.

IBM1051I I **Argument to** *BUILTIN name* **built-in may not be byte aligned.**

Explanation: This message applies to the ADDR, CURRENTSTORAGE/SIZE and STORAGE/SIZE built-in functions. Applying any one of these built-in functions to an unaligned bit variable may not produce the results you expected.

© Copyright IBM Corp. 1999, 2018

IBM1052I I The NODESCRIPTOR attribute is accepted even though some arguments have * extents.

Explanation: When a string with * extent or an array with * extents is passed, PL/I normally passes a descriptor so that the called routine knows how big the passed argument really is. The NODESCRIPTOR attribute indicates that no descriptor should be passed; this is invalid if the called routine is a PL/I procedure.

```
dcl x entry( char(*), fixed bin(31) )
      options( nodescriptor );
```

IBM1053I I Scaled FIXED operation evaluated as FIXED DECIMAL.

Explanation: If one of the built-in functions ADD, DIVIDE, MULTIPLY or SUBTRACT is invoked with argument that have type FIXED, if either operand has a non-zero scale factor, the result will have type FIXED DEC.

IBM1058I I Conversion from *source type* to *target type* will be done by library call.

Explanation: This message can be used to help find code that may be very expensive if executed as part of a loop or to find code involving conversions of unlike types.

IBM1059I I SELECT statement contains no OTHERWISE clause.

Explanation: The ERROR condition will be raised if no WHEN clause is satisfied.

IBM1060I I Name resolution for *identifier* selected its declaration in a structure, rather than its non-member declaration in a parent block.

Explanation: The PL/I language rules require this, but it might be a little surprising. In the following code fragment, for instance, the display statement would display the value of x.y.

```
a: proc;

  dcl y fixed bin init(3);

  call b;

  b: proc;

    dcl
      1 x,
        2 y fixed bin init(5),
        2 z fixed bin init(7);

      display( y );

    end;

  end a;
```

IBM1061I I Probable DATE calculation should be examined for validity after the year 1999.

Explanation: Use of any of the constants 365, 1900 or '19' may indicate a date calculation. If this is true, you should examine the calculation to determine if it will be valid after the year 1999.

IBM1062I I *variable* inferred to contain a two-digit year.

Explanation: The indicated was inferred to contain a two-digit year because, for example, it was assigned the DATE built-in function.

IBM1063I I Code generated for DO group would be more efficient if control variable were a 4-byte integer.

Explanation: The control variable in the DO loop is a 1-byte integer, 2-byte integer, fixed decimal or fixed picture, and consequently, the code generated for the loop will not be optimal.

IBM1064I I Use of OPT(2) forces TEST(BLOCK).

Explanation: Under OPT(2), any specification of TEST hooks stronger than TEST(BLOCK) is not supported.

IBM1065I I Float constant *constant* would be more precise if specified as a long float.

Explanation: The named short floating-point constant cannot be exactly represented. It could be more accurately represented if it were specified as a long floating-point constant. For example, the 1.3E0 cannot be exactly represented, but could be better represented as 1.3D0.

IBM1067I I UNTIL clause ignored.

Explanation: If a DO specification has no clause such as TO, BY or REPEAT that could cause the loop to be repeated, then the UNTIL clause will have no effect on the loop and will be ignored.

```
do x = y until ( z > 0 );
  ...
end;
```

IBM1068I I Procedure has no RETURNS attribute, but contains a RETURN statement. A RETURNS attribute will be assumed.

Explanation: If a procedure contains a RETURN statement, it should have the RETURNS attribute specified on its PROCEDURE statement.

```
a: proc;
  return( 0 );
end;
```

IBM1069I I The AUTOMATIC variables in a block should not be used in the prologue of that block.

Explanation: The AUTOMATIC variables in a block may be used in the declare statements and the executable statements of any contained block, but in the block in which they are declared, they should be used only in the executable statements.

```
dcl x fixed bin(15) init(5);
dcl y(x) fixed bin(15);
```

IBM2800I I The procedure *proc name* **is not referenced.**

Explanation: The named procedure is not external and is never referenced in the compilation unit. This may represent an error (if it was supposed to be called) or an opportunity to eliminate some dead code.

IBM2801I I FIXED DEC(*source-precision***,***source-scale***) operand will be converted to FIXED BIN(***target-precision***,***target-scale***). This introduces a non-zero scale factor into an integer operation and will produce a result with the attributes FIXED BIN(***result-precision***,***result-scale***).**

Explanation: Under RULES(IBM), when an arithmetic operation has an operand that is FIXED BIN and an operand that is FIXED DEC with a non-zero scale factor, then the FIXED DEC operand will be converted to FIXED BIN.

IBM2802I I Aggregate mapping will be done by library call.

Explanation: This message can be used to help find code that may be very expensive if executed as part of a loop. It may be produced, for example, if your code refers to an element of a structure that uses REFER. If the structure uses multiple REFERs and the element occurs after the last REFER, the single reference to that element may produce multiple copies of this message (because multiple library calls will be made).

IBM2803I I *keyword* **STRING EDIT statement optimized.**

Explanation: This message is issued when a PUT or GET STRING EDIT statement has been optimized by the compiler so that most of it is done inline.

IBM2804I I Boolean is compared with something other than '1'b or '0'b.

Explanation: This message will flag statements such as the following, where "true" is a BIT(1) STATIC INIT('1'b). It would be better if "true" were a named constant, i.e. if it were declared with the VALUE attribute rather than STATIC INIT

```
if ( a < b ) = true then
```

IBM2805I I For assignment to *variable name*, **conversion from** *source type* **to** *target type* **will be done by library call.**

Explanation: This message can be used to help find code that may be very expensive if executed as part of a loop or to find code involving conversions of unlike types.

IBM2806I I Passing a LABEL to another routine is poor coding practice and will cause the compiler to generate less than optimal code.

Explanation: It is generally very unwise to pass a label to another routine. It would be good to think about redesigning any code doing this.

IBM2809I I FIXED DEC(*source-precision***,***source-scale***) operand will be converted to FIXED BIN(***target-precision***,***target-scale***). This introduces 8-byte integer arithmetic into an operation that might be faster if computed in decimal.**

Explanation: If the LIMITS option specifies a maximum FIXED precision greater than 31, then an operation involving a FIXED DEC and a FIXED BIN operand might produce an 8-byte integer result even if both operands are "small". For example, if you add a FIXED DEC(13) and a FIXED BIN(31), the result would be an 8-byte integer (because a FIXED DEC(13) value might be too large to fit in a 4-byte integer). To avoid this, you could apply the DECIMAL built-in function to the FIXED BIN operand.

IBM2810I I Conversion of FIXED BIN(*source-precision***,***source-scale***) to FIXED DEC(***target-precision***,***target-scale***) may produce a more accurate result than under the old compiler.**

Explanation: In certain conversions of FIXED BIN(p,q) to FIXED DEC, the old compiler slightly rounded the result if q was positive.

IBM2811I I Use of PICTURE as DO control variable is not recommended.

Explanation: If the control variable in a DO loop is a PICTURE variable, then more code will be generated for the loop than if the control variable were a FIXED BIN variable. Moreover, such loops may easily be miscoded so that they will loop infinitely.

IBM2812I I Argument number *argument number* **to** *BUILTIN name* **built-in would lead to much better code if declared with the VALUE attribute.**

Explanation: For functions such as VERIFY(x,y), if y is a constant, it is much better for performance to declare y with the VALUE attribute rather than with the INITIAL attribute.

IBM2814I I Aggregate mapping for storage allocation will be done by library call.

Explanation: This message can be used to help find code that may be expensive if invoked many times. This message may be produced for ALLOCATE statements for BASED and CONTROLLED variables with non-constant extents, and it may also be produced for the prologue of PROCEDUREs that use AUTOMATIC variables with non-constant extents.

IBM2815I I Argument number *argument-number* **in ENTRY reference** *ENTRY name* **is not recommended to be passed BYVALUE.**

Explanation: A BYVALUE argument should be one that could reasonably be passed in a register. Hence its type should be either one of REAL FIXED BIN, REAL FLOAT, POINTER, OFFSET, HANDLE, LIMITED ENTRY, FILE, ORDINAL, CHAR(1), WCHAR(1), or ALIGNED BIT(n) with n less than or equal to 8.

IBM2816I I BYVALUE parameters should ideally be ones that can reasonably be passed in registers.

Explanation: A BYVALUE parameter should be one that could reasonably be passed in a register. Hence its type should be either one of REAL FIXED BIN, REAL FLOAT, POINTER, OFFSET, HANDLE, LIMITED ENTRY, FILE, ORDINAL, CHAR(1), WCHAR(1), or ALIGNED BIT(n) with n less than or equal to 8.

IBM2817I I BYVALUE in RETURNS is recommended only for types that can reasonably be returned in registers.

Explanation: Using BYVALUE in RETURNS is recommended only if the value to be returned has a type that could reasonably be returned in a register. Hence its type should be either one of REAL FIXED BIN, REAL FLOAT, POINTER, OFFSET, HANDLE, LIMITED ENTRY, FILE, ORDINAL, CHAR(1), WCHAR(1), or ALIGNED BIT(n) with n less than or equal to 8.

IBM2818I I Addition or subtraction of FIXED DEC(*precision***,***scale-factor***) and FIXED DEC(***precision***,***scale-factor***) may raise FIXEDOVERFLOW.**

Explanation: The precision required to hold the result as defined by PL/I of this add (or subtract) is greater than the LIMITS(FIXEDDEC) maximum for the operands and hence depending on the data values, FIXEDOVERFLOW may be raised by the operation.

IBM2819I I Multiplication of FIXED DEC(*precision***,***scale-factor***) and FIXED DEC(***precision***,***scale-factor***) may raise FIXEDOVERFLOW.**

Explanation: The precision required to hold the result as defined by PL/I of this multiply is greater than the LIMITS(FIXEDDEC) maximum for the operands and hence depending on the data values, FIXEDOVERFLOW may be raised by the operation.

IBM2820I I The *option-name* **option is not supported on this platform.**

Explanation: The named compiler option is not supported on this platform. For example, the BLKOFF option is an option on the z/OS platform, but not on AIX or Windows. If specified on those platforms, it is ignored.

IBM2825I I Conversion from *source type* **to** *target type* **will be done by library call.**

Explanation: This message can be used to help find code that may be very expensive if executed as part of a loop or to find code involving conversions of unlike types.

IBM2826I I For assignment to *variable name***, conversion from** *source type* **to** *target type* **will be done by library call.**

Explanation: This message can be used to help find code that may be very expensive if executed as part of a loop or to find code involving conversions of unlike types.

IBM2827I I **Conversion from** *source type* **to** *target type* **can produce an inexact or incorrect result.**

Explanation: For example, the conversion of the FLOAT DEC(15) value 321.1234 to FIXED DEC(15,15) will produce the inexact result 0.123399999999952. However, the conversion of the FLOAT DEC(15) value 54321.1234 to FIXED DEC(15,15) will produce the incorrect result 0.372036854775807. Incorrect results can be avoided in a conversion to FIXED DEC(p,q) if the absolute value of the source is less than 10**(18-p).

IBM2830I I **VALUE(:** *type name* **:) will return an instance of the structure type that is only partially initialized.**

Explanation: If the VALUE type function is applied to a structure type which has an initial attribute on only some of its elements, then the structure instance will be only partially initialized. For example, the compiler will flag the following code with this message because B2 has no initial value - it will have the initial values from type a only if B2 is also declared with the attribute init(value(: a :)).

```
define structure
  1 a,
    2 a1 fixed bin(31) init( 17 ),
    2 a2 fixed bin(31) init( 19 );

define structure
  1 b,
    2 b1 fixed bin(31) init( 119 ),
    2 b2 type a;

dcl x type b;

x = value(: b :);
```

IBM2831I I **ASSERT statement may never be executed.**

Explanation: This message warns that the compiler has detected an ASSERT UNREACHABLE statement that can never be run as the flow of control must always pass it by.

IBM2832I I **INLINE directive will be ignored for** *procedure name* **and all other procedures since the TEST option is on.**

Explanation: The compiler will perform no inling if the TEST option is on.

IBM2833I I **INLINE directive will be ignored for** *procedure name* **since it contains ENTRY statements.**

Explanation: The compiler will not inline a PROCEDURE that has ENTRY statements.

IBM2834I I **INLINE directive will be ignored for** *procedure name* **since it contains subprocedures and/or BEGIN blocks.**

Explanation: The compiler will not inline a PROCEDURE or BEGIN block that contains other PROCEDUREs or BEGIN blocks.

IBM2835I I **INLINE directive will be ignored for** *procedure name* **since it has OPTIONS(NODESCRIPTOR), but has some parameters with nonconstant extents.**

Explanation: The compiler will not inline a PROCEDURE that requires has the NODESCRIPTOR option but would normally be passed descriptors with its arguments.

IBM2836I I **INLINE directive will be ignored for** *procedure name* **since it contains labels that may be targets of out-of-block GOTOs.**

Explanation: The compiler will not inline a PROCEDURE that which has any labels that are possibly the target of a GOTO from another PROCEDURE or BEGIN block.

IBM2837I I **INLINE directive will be ignored for** *procedure name* **since it contains some DATA-directed I/O statements.**

Explanation: The compiler will not inline a PROCEDURE that has any PUT DATA or GET DATA statements.

IBM2838I I **INLINE directive will be ignored for** *procedure name* **since it has non-default condition enablement.**

Explanation: The compiler will not inline a PROCEDURE that has any condition enablement that differs from the default.

IBM2839I I **INLINE directive will be ignored for** *procedure name* **since it contains ON-units.**

Explanation: The compiler will not inline a PROCEDURE that has any ON statementes.

IBM2840I I **If TRANSLATE is being used to reformat a date-time value, it would be better to use the REPATTERN or DATETIME built-in function instead.**

Explanation: If the first and third arguments to the TRANSLATE built-in function are both constant, then the code is likely trying to reformat a date-time value. This code would be easier to understand if the

REPATTERN built-in function or, if possible, the DATETIME built-in function were used instead. For example, the first two bits of code below assign the same value to the target variable shortdate, and the second two bits of code also assign the same value to the target variable currentdate. However, in each case, the second statement is much clearer.

```
shortdate
  = translate( '12.34.5678',
               longdate,
               '56783412abcdefghijkl' );

shortdate
 = repattern( longdate,
              'DD.MM.YYYY',
              'YYYYMMDDHHMISS999' );

currentdate
  = translate( '12.34.5678',
               datetime(),
               '56783412abcdefghijkl' );

currentdate
 = datetime( 'DD.MM.YYYY' );
```

IBM2841I I **Changing MEMCONVERT(p,n,1200,q,m,1208) to MEMCU12(p,n,q,m) would be better for performance.**

Explanation: MEMCU12 will perform much better than MEMCONVERT.

IBM2842I I **Changing MEMCONVERT(p,n,1208,q,m,1200) to MEMCU21(p,n,q,m) would be better for performance.**

Explanation: MEMCU21 will perform much better than MEMCONVERT.

IBM2843I I **The defined structure** *struct name* **is** *alignment* **byte aligned, but occupies only** *storage size* **bytes of storage.**

Explanation: Defined structures must occupy a number of bytes that is a multiple of the structure's alignment. So, for example, if a structure contains an aligned fixed bin(31) (or other aligned fullword) field as its most stringently aligned item, then the structure must occupy a multiple of 4 bytes. The following structure does not meet this requirement:

```
define structure
  1 point,
    2 x  fixed bin(31),
    2 y  char(1);
```

IBM2844I I **The characters =+ will be accepted as two separate characters. But perhaps += was meant, and it would be better to separate these characters with a blank.**

Explanation: This may represent a problem especially if this occurs in an assignment statement and += was meant instead of =+.

IBM2845I I **The characters =- will be accepted as two separate characters. But perhaps -= was meant, and it would be better to separate these characters with a blank.**

Explanation: This may represent a problem especially if this occurs in an assignment statement and -= was meant instead of =-.

IBM2846I I **It would be better to convert nested procedures in a PACKAGE into sister level-1 procedures.**

Explanation: The compiler issues this message if a compilation unit contains a PACKAGE statement with exactly one level-1 procedure which in turn has its own nested procedures.

Chapter 2. Compiler Warning Messages (1078-1225, 2600-2799)

IBM1078I W Statement may never be executed.

Explanation: This message warns that the compiler has detected a statement that can never be run as the flow of control must always pass it by.

IBM1079I W Too few arguments have been specified for the ENTRY *ENTRY name***.**

Explanation: The number of arguments should match the number of parameters in the ENTRY declaration.

IBM1080I W The keyword *label-name***, which could form a complete statement, is accepted as a label name, but a colon may have been used where a semicolon was meant.**

Explanation: A PL/I keyword which could form a complete statement has been used as statement label. This usage is accepted, but a colon may have been used where a semicolon was intended.

```
dcl a fixed bin(31) ext;

if a = 0 then
  put skip list( 'a = 0' )
else:

a = a + 1;
```

IBM1081I W *keyword* **expression should be scalar. Lower bounds assumed for any missing subscripts.**

Explanation: The expression in the named keyword clause should be a scalar, but an array reference was specified.

```
dcl p     pointer;
dcl x     based char(10);
dcl a(10) area(1000);

allocate x in(a) set(p);
```

IBM1082I W Argument number *argument-number* **in entry reference** *entry name* **is a scalar, but its declare specifies a structure.**

Explanation: A scalar may be passed as the argument when a structure is expected, but this require building a "dummy" structure and assigning the scalar to each field in that structure.

```
dcl e entry( 1 2 fixed bin(31), 2 fixed bin(31) );
dcl i fixed bin(15);
call e( i );
```

IBM1083I W Source in label assignment is inside a DO-loop, and an illegal jump into the loop may be attempted. Optimization will also be very inhibited.

Explanation: GOTO statements may not jump into DO loops, and the compiler will flag any GOTO whose target is a label constant inside a (different) DO loop. However, if a label inside a DO loop is assigned to a label variable, then this kind of error may go undetected.

IBM1084I W Nonblanks after right margin are not allowed under RULES(NOLAXMARGINS).

Explanation: Under RULES(NOLAXMARGINS), there should be nothing but blanks after the right margin.

IBM1085I W *variable* **may be unset when used.**

Explanation: The indicated variable may not have been assigned or initialized a value before it is used.

IBM1086I W *built-in function* **will be evaluated using long rather than extended routines.**

Explanation: The indicated built-in function has an extended float argument, but since the corresponding extended routine is not yet available, it will be evaluated using the appropriate long routine.

IBM1087I W FLOAT source is too big for its target. An appropriate HUGE value of *assumed value* **is assumed.**

Explanation: A value larger than HUGE(1s0) cannot be assigned to a short float. Under hexadecimal float, the value 3.141592E+40 could be assigned to a short float, but under IEEE, the maximum value that a short float can hold is about 3.40281E+38.

IBM1088I W FLOAT literal is too big for its implicit precision. The E in the exponent will be replaced by a D.

Explanation: The precision for a float literal is implied by the number of digits in its mantissa. For instance

© Copyright IBM Corp. 1999, 2018

1e99 is implicitly FLOAT DECIMAL(1), but the value 1e99 is larger than the largest value a FLOAT DECIMAL(1) can hold.

IBM1089I W Control variable in DO loop cannot exceed TO value, and loop may be infinite.

Explanation: If the TO value is equal to the maximum value that a FIXED or PICTURE variable can hold, then a loop dominated by that variable will run endlessly unless exited inside the loop by a LEAVE or GOTO. For example, in the first code fragment below, x can never be bigger than 99, and the loop would be infinite. In the second code fragment below, y can never be bigger than 32767, and the loop would be infinite.

```
dcl x pic'99';

do x = 1 to 99;
  put skip list( x );
end;

dcl y fixed bin(15);

do y = 1 to 32767;
  put skip list( y );
end;
```

IBM1090I W Constant used as locator qualifier.

Explanation: An expression contains a reference to a based variable with a constant value for its locator qualifier. This may cause a protection exception on some systems. It may also indicate that the variable was declared as based on NULL or SYSNULL and that this constant value is being used as its locator qualifier.

```
dcl a fixed bin(31) based( null() );

a = 0;
```

IBM1091I W FIXED BIN precision less than storage allows.

Explanation: Except in unusual circumstances, the precision in a FIXED BIN declaration should be 7, 15, 31 or 63 if SIGNED and one greater if UNSIGNED. This message may indicate that a declare specified, for example, FIXED BIN(8) when UNSIGNED FIXED BIN(8) was meant.

IBM1092I W GOTO whose target is or may be in another block severely limits optimization.

Explanation: Try to change the code so that it sets and tests a switch instead, or limit GOTOs to very small modules that do not need optimization.

IBM1093I W PLIXOPT string is invalid. See related runtime message *message-number***.**

Explanation: The PLIXOPT string could not be parsed. See the cited LE message for more detail.

IBM1094I W Element *option* **in PLIXOPT is invalid. See related runtime message** *message-number***.**

Explanation: The PLIXOPT string contains an invalid item. See the cited LE message for more detail.

IBM1095I W Element *option* **in PLIXOPT has been remapped to** *option***. See related runtime message** *message-number***.**

Explanation: The PLIXOPT string contains a run-time option which is not supported by LE. See the cited LE message for more detail.

IBM1096I W STAE and SPIE in PLIXOPT is not supported. See related runtime message *message-number***.**

Explanation: The SPIE and STAE options have been replaced by the TRAP option. TRAP(ON) is equivalent to SPIE and STAE; TRAP(OFF) is equivalent to NOSPIE and NOSTAE. The combination SPIE and NOSTAE and the combination NOSPIE and STAE are no longer supported. See the cited LE message for more detail.

IBM1097I W Scalar accepted as argument number *argument-number* **in ENTRY reference** *ENTRY name* **although parameter description specifies an array.**

Explanation: Generally, scalars should not be passed where arrays are expected, but in some situations, this may be what you want.

```
dcl a entry( (*) fixed bin ) option(nodescriptor);

call a( 0 );
```

IBM1098I W Extraneous comma at end of statement ignored.

Explanation: A comma was followed by a semicolon rather than by a valid syntactical element (such as an identifier). The comma will be ignored in order to make the semicolon valid.

```
dcl 1 a, 2 b fixed bin, 2 c fixed bin, ;
```

IBM1099I W FIXED DEC(*source-precision*,*source-scale*) operand will be converted to FIXED BIN(*target-precision*,*target-scale*). Significant digits may be lost.

Explanation: Under RULES(IBM), when a comparison or arithmetic operation has an operand that is FIXED BIN and an operand that is FIXED DEC with a non-zero scale factor, then the FIXED DEC operand will be converted to FIXED BIN. Under RULES(ANS), when a comparison or arithmetic operation has an operand that is FIXED BIN and an operand that is FIXED DEC with a zero scale factor, then the FIXED DEC operand will be converted to FIXED BIN. In each case, significant digits may be lost, and if there is a fractional part, it may not be exactly represented as binary. For instance, under RULES(IBM), the assignment statement below will cause the target to have the value 29.19, and in the comparison, C will be converted to FIXED BIN(31,10) and significant digits will be lost (in fact, SIZE would be raised, but since it is disabled, this program would be in error).

```
dcl a fixed dec(07,2) init(12.2);
dcl b fixed bin(31,0) init(17);
dcl c fixed dec(15,3) init(2097151);
dcl d fixed bin(31,0) init(0);

a = a + b;

if c = d then;
```

IBM1100I W The attribute *attribute-option* is not valid on BEGIN blocks and is ignored.

Explanation: An attribute (REDUCIBLE in the example below) has been specified in the OPTIONS clause on a BEGIN statement, but that attribute is not valid for BEGIN blocks.

```
begin options( reducible );
```

IBM1101I W *option-name* is not a known PROCEDURE attribute and is ignored.

Explanation: An attribute (DATAONLY in the example below) has been specified in the OPTIONS clause on a PROCEDURE statement, but that attribute is not valid for PROCEDUREs.

```
a: proc options( dataonly );
```

IBM1102I W *option-name* is not a known BEGIN attribute and is ignored.

Explanation: The indicated attribute is valid on PROCEDURE statements, but not on BEGIN statements.

```
begin recursive;
```

IBM1103I W *option-name* is not a supported compiler option and is ignored.

Explanation: The compiler option is not supported on this platform.

```
*process map;
```

IBM1104I W Suboptions of the compiler option *option-name* are not supported and are ignored.

Explanation: Suboptions of the compiler option are not supported on this platform.

```
*process list(4);
```

IBM1105I W A suboption of the compiler option *option-name* is too long. It is shortened to length *number-of-letters*.

Explanation: Various compiler options have limits on the size of subfields. Refer to the Programming Guide for the limits of specific compiler options.

```
*process margini( '+-' );
```

IBM1106I W Condition prefixes on *keyword* statements are ignored.

Explanation: Condition prefixes are not allowed on DECLARE, DEFAULT, IF, ELSE, DO, END, SELECT, WHEN or OTHERWISE statements.

```
(nofofl): if (x+y) > 0 then
```

IBM1107I W *option-name* is not a known ENTRY statement attribute and is ignored.

Explanation: An attribute (DATAONLY in the example below) has been specified in the OPTIONS clause on an ENTRY statement, but that attribute is not valid for ENTRY statements.

```
a: entry options( dataonly );
```

IBM1108I W The character *char* specified in the *option* option is already defined and may not be redefined. The redefinition will be ignored.

Explanation: A character specified in the OR, NOT, QUOTE or NAMES compiler option is already defined in the PL/I character set or by another compiler option.

```
*process not('=');
*process not('!') or('!');
```

IBM1109I W The second argument in the C-format item will be ignored.

Explanation: If you wish to display the real and imaginary parts of a complex number using different formats, use the REAL and IMAG built-in functions and 2 format items.

```
put edit ( x ) ( c( e(10,6), e(10,6) ) );
```

IBM1110I W The %INCLUDE statement should be on a line by itself. The source on the line after the %INCLUDE statement is ignored.

Explanation: Split the text into 2 lines.

```
%include x; %include y;
```

IBM1111I W CHECK prefix is not supported and is ignored.

Explanation: The CHECK prefix is not part of the SAA PL/I language.

```
(check): i = j + 1;
```

IBM1112I W *condition-name* **condition is not supported and is ignored.**

Explanation: The CHECK and PENDING conditions are not part of the SAA PL/I language.

```
on check ...
```

IBM1113I W *verb-name* **statement is not supported and is ignored.**

Explanation: The named statement, for example the CHECK statement, is not part of the SAA PL/I language.

IBM1114I W Comparands are both constant.

Explanation: Both operands in a comparison are constant, and consequently, the result of the comparison is also a constant. If this comparison is the expression in an IF clause, for example, this means that either the THEN or ELSE clause will never be executed.

IBM1115I W INITIAL list contains *count* **items, but the array** *variable name* **contains only** *array size***. Excess is ignored.**

Explanation: For an array, an INITIAL list should not contain more values than the array has elements.

```
dcl a init( 1, 2 ), b(5) init( (10) 0 );
```

IBM1116I W Comment spans more than one file.

Explanation: A comment ends in a different file than it begins. This may indicate that an end-of-comment statement is missing.

IBM1117I W String spans more than one file.

Explanation: A string ends in a different file than it begins. This may indicate that a closing quote is missing.

IBM1118I W Delimiter missing between *nondelimiter* **and** *nondelimiter***. A blank is assumed.**

Explanation: A delimiter (for example, a blank or a comma) is required between all identifiers and constants.

```
dcl 1 a, 2 b, 3c;
```

IBM1119I W Code generated for DO group would be more efficient if control variable *name* **were not an aggregate member.**

Explanation: The control variable in the DO loop is a member of an array, a structure or a union, and consequently, the code generated for the loop will not be optimal.

IBM1120I W Multiple closure of groups. END statements will be inserted to close intervening groups.

Explanation: Using one END statement to close more than one group of statements is permitted, but it may indicate a coding error.

IBM1121I W Missing *character* **assumed.**

Explanation: The indicated character is missing, and there are no more characters in the source. The missing character has been inserted by the parser in order to correct your source.

IBM1122I W Missing *character* assumed before *character*.

Explanation: The indicated character is missing and has been inserted by the parser in order to correct your source.

```
display( 'Program starting' ;
```

IBM1123I W The ENVIRONMENT option *option-name* has been specified without a suboption. The option *option-name* is ignored.

Explanation: Certain ENVIRONMENT options, such as RECSIZE, require suboptions.

```
dcl f file env( recsize );
```

IBM1124I W A suboption has been specified for the ENVIRONMENT option *option-name*. The suboption will be ignored.

Explanation: Certain ENVIRONMENT options, such as CONSECUTIVE, should be specified without any suboptions.

```
dcl f file env( consecutive(1) );
```

IBM1125I W The ENVIRONMENT option *option-name* has been specified more than once.

Explanation: ENVIRONMENT options should not be repeated.

```
dcl f file env( consecutive consecutive );
```

IBM1126I W The ENVIRONMENT option *option-name* has an invalid suboption. The option will be ignored.

Explanation: The suboption type is incorrect.

```
dcl f file env( regional(5) );
```

IBM1127I W *option-name* is not a known ENVIRONMENT option. It will be ignored.

Explanation: There is no such supported ENVIRONMENT option.

```
dcl f file env( unknown );
```

IBM1128I W The ENVIRONMENT option *option-name* conflicts with the LANGLVL compiler option. The option will be ignored.

Explanation: The indicated option is valid only with LANGLVL(OS).

```
dcl f file env( fb );
```

IBM1129I W *verb-name processor-name* statement ignored up to closing semicolon.

Explanation: An EXEC SQL or EXEC CICS statement has been found in the source program. The compiler will ignore these statements.

```
exec sql ...;
```

IBM1130I W The external name *identifier* is too long. It will be shortened to *identifier*.

Explanation: The maximum length of external names is set by the EXTNAME suboption of the LIMITS compiler option.

```
dcl this_name_is_long  static external pointer;
```

IBM1131I W An EXTERNAL name specification for *name* has been specified on its PROCEDURE statement and in the EXPORTS clause of the PACKAGE statement. The EXPORTS specification will be used.

Explanation: The name specified in the EXTERNAL attribute in the EXPORTS clause overrides the name specified in the EXTERNAL attribute on the PROCEDURE statement.

```
a: package exports( b ext('_B') );

  b: proc  ext( 'BB' );
```

IBM1132I W An EXTERNAL name specification for *name* has been specified in its declaration and in the RESERVES clause of the PACKAGE statement. The RESERVES specification will be used.

Explanation: The name specified in the EXTERNAL attribute in the RESERVES clause overrides the name specified in the EXTERNAL attribute in the DECLARE statement.

```
a: package reserves( b ext('_B') );

  dcl b ext( 'BB' ) static ...
```

IBM1133I W The FORMAT CONSTANT array *label-name* is not fully initialized.

Explanation: An element of a FORMAT CONSTANT array has not been defined, for example, f(2) in the example below.

```
f(1): format( x(2), a );

f(3): format( x(4), a );
```

IBM1134I W The LABEL CONSTANT array *label-reference* is not fully initialized.

Explanation: The named variable defines a statement label array, but not all the elements in that array are labels for statements in the containing procedure.

```
l(1): display( ... );

l(3): display( ... );
```

IBM1135I W Logical operand is constant.

Explanation: An argument to one of the logical operators (or, and or not) is a constant. The result of the operation may also be a constant. If this operation is the expression in an IF clause, for example, this means that either the THEN or ELSE clause will never be executed.

```
if a | '1'b then
```

IBM1136I W Function invoked as a subroutine.

Explanation: A function, for example, a PROCEDURE or ENTRY statement with the RETURNS attribute, has been invoked in a CALL statement. The value that is returned by the function will be discarded, but the OPTIONAL attribute should be used to indicate that this is valid.

IBM1137I W The attribute *attribute* is invalid in GENERIC descriptions and will be ignored.

Explanation: The named attribute is invalid in GENERIC description lists.

```
dcl g generic ( f1 when( connected ),
                f2 otherwise );
```

IBM1138I W Number of items in INITIAL list is *count* for the array *variable name* which contains *array size* elements.

Explanation: The array will be incompletely initialized. If the named variable is part of a structure, subsequent elements in that structure with this problem will be flagged with message 2602. This may be a programming error (in the example below, 4 should probably have been 6) and may cause exceptions when the program is run.

```
dcl a(8) fixed dec init( 1, 2, (4) 0 );
```

IBM1139I W Syntax of the %CONTROL statement is incorrect.

Explanation: The %CONTROL statement must be followed by FORMAT or NOFORMAT option enclosed in parentheses and then a semicolon.

IBM1140I W Syntax of the LANGLVL option in the %OPTION statement is incorrect.

Explanation: The LANGLVL option in the %OPTION statement must be specified as either LANGLVL(SAA) or LANGLVL(SAA2).

IBM1141I W Syntax of the %NOPRINT statement is incorrect.

Explanation: The %NOPRINT statement must be followed, with optional intervening blanks, by a semicolon.

IBM1142I W Syntax of the %PAGE statement is incorrect.

Explanation: The %PAGE statement must be followed, with optional intervening blanks, by a semicolon.

IBM1143I W Syntax of the %PRINT statement is incorrect.

Explanation: The %PRINT statement must be followed, with optional intervening blanks, by a semicolon.

IBM1144I W Number of lines specified with %SKIP must be between 0 and 999 inclusive.

Explanation: Skip amounts greater than 999 are not supported.

```
%skip(2000);
```

IBM1145I W Syntax of the %SKIP statement is incorrect.

Explanation: The %SKIP statement must be followed by a semicolon with optional intervening blanks and a parenthesized integer.

IBM1146I W Syntax of the TEST option in the %OPTION statement is incorrect.

Explanation: The TEST option in the %OPTION statement must be specified without any suboptions.

IBM1147I W Syntax of the NOTEST option in the %OPTION statement is incorrect.

Explanation: The NOTEST option in the %OPTION statement must be specified without any suboptions.

IBM1148I W Syntax of the %PUSH statement is incorrect.

Explanation: The %PUSH statement must be followed, with optional intervening blanks, by a semicolon.

IBM1149I W Syntax of the %POP statement is incorrect.

Explanation: The %POP statement must be followed, with optional intervening blanks, by a semicolon.

IBM1150I W Syntax of the %NOTE statement is incorrect.

Explanation: The %NOTE statement must be followed by, in parentheses, a note and an optional return code, and then a semicolon.

IBM1151I W FIXED BINARY precision is reduced to *maximum value***.**

Explanation: The maximum FIXED BIN precision depends on the LIMITS option.

IBM1152I W FIXED DECIMAL precision is reduced to *maximum value***.**

Explanation: The maximum FIXED DEC precision depends on the LIMITS option.

IBM1153I W FLOAT BINARY precision is reduced to *maximum value***.**

Explanation: The maximum FLOAT BIN precision is 64 on Intel, 106 on AIX and 109 on z/OS.

IBM1154I W FLOAT DECIMAL precision is reduced to *maximum value***.**

Explanation: The maximum FLOAT DEC precision is 18 on Intel, 32 on AIX and 33 on z/OS except for DFP which has a maximum of 34.

IBM1155I W The aggregate *aggregate-name* **contains noncomputational values. Those values will be ignored.**

Explanation: Some members of an aggregate referenced in an I/O statement are noncomputational. The computational members will be correctly processed, but the noncomputational ones will be ignored.

```
dcl 1 x,
      2 y ptr,
      3 fixed bin(31);
put skip list(x);
```

IBM1156I W Arguments to MAIN procedure are not all POINTER.

Explanation: Under SYSTEM(CICS), SYSTEM(TSO) and SYSTEM(IMS), the arguments to the MAIN procedure should all have type POINTER.

IBM1157I W *note*

Explanation: This message is used by %NOTE statements with a return code of 4.

IBM1158I W A *option* **is missing in the specification of the** *option* **option. One is assumed.**

Explanation: A closing quote or parenthesis is missing in the specification of a compiler option. A quoted string must not cross line boundaries.

IBM1159I W The string *option* **is not recognized as a valid option keyword and is ignored.**

Explanation: An invalid compiler option has been specified.

IBM1160I W The third argument to the MARGINS option is not supported.

Explanation: Printer control characters are not supported on input source records.

IBM1161I W The suboption *suboption* **is not valid for the** *option* **compiler option.**

Explanation: A suboption of a compiler option is incorrect. The suboption may be unknown or outside the allowable range.

```
*process flag(q)  margins(1002);
```

IBM1162I W A required suboption is missing for the *suboption* option.

Explanation: A required suboption of a compiler option is missing.

```
*process or;
```

IBM1163I W Required sub-fields are missing for the *option* option. Default values are assumed.

Explanation: Required suboptions of a compiler option are missing.

```
*process margins;
```

IBM1164I W *option-name* should be specified within OPTIONS, but is accepted as is.

Explanation: The option, for example REORDER, is accepted outside of the OPTIONS attribute, but it should be specified within the OPTIONS attribute. This would also conform to the ANSI standard.

IBM1165I W The OPTIONS option *option-name* has been specified more than once.

Explanation: The only supported LINKAGE options are OPTLINK and SYSTEM.

IBM1166I W *option-name* is not a known LINKAGE suboption. The LINKAGE option will be ignored.

Explanation: The only supported LINKAGE options are OPTLINK and SYSTEM.

IBM1167I W Maximum number of %PUSH statements exceeded. The control statement is ignored.

Explanation: The maximum number of pending %PUSH statements is 63.

IBM1168I W No %PUSH statements are in effect. The %POP control statement is ignored.

Explanation: A %POP has been issued when no %PUSH statement are pending.

IBM1169I W No precision was specified for the result of the *builtin name* built-in. The precision will be determined from the argument.

Explanation: This message applies to the FIXED and FLOAT built-in functions when only one argument is given. The precision is not set to a default, but is instead derived from the argument. For example, if x is FLOAT BIN(21), FIXED(x) will return a FIXED BIN(21) value.

IBM1170I W The OPTIONS attribute *option-attribute* is not supported and is ignored.

Explanation: The indicated element of the OPTIONS list is not supported.

```
dcl a ext entry options( nomap );
```

IBM1171I W SELECT statement contains no WHEN or OTHERWISE clauses.

Explanation: WHEN or OTHERWISE clauses are not required on SELECT statements, but their absence may indicate a coding error.

IBM1172I W A zero length string has been entered for the *option-name* option. The option is ignored.

Explanation: User-specified string has zero length. This can occur when OR('') has been specified on the command line or when the backslash character is specified as the only character in the OR string. In the latter case, the backslash character has been interpreted as an escape character, and so the string appears to have zero length.

IBM1173I W SELECT statement contains no WHEN clauses.

Explanation: SELECT statements do not require WHEN clauses, but their absence may indicate a coding error.

IBM1174I W The reference in the *from-into clause* clause may not be byte-aligned.

Explanation: The reference specified in the FROM or INTO clause may not be byte-aligned. If the reference is indeed not byte-aligned, unpredictable results may occur.

IBM1175I W FIXED BINARY constant contains too many digits. Excess nonsignificant digits will be ignored.

Explanation: The maximum precision for FIXED BINARY constants is specified by the FIXEDBIN suboption of the LIMITS compiler option.

IBM1176I W FIXED DECIMAL constant contains too many digits. Excess nonsignificant digits will be ignored.

Explanation: The maximum precision for FIXED DECIMAL constants is specified by the FIXEDDEC suboption of the LIMITS compiler option.

IBM1177I W Mantissa in FLOAT BINARY constant contains more digits than the implementation maximum. Excess nonsignificant digits will be ignored.

Explanation: Float binary constants are limited to 64 digits on Intel, 32 on AIX and 33 on z/OS.

IBM1178I W Mantissa in FLOAT DECIMAL constant contains more digits than the implementation maximum. Excess nonsignificant digits will be ignored.

Explanation: Float decimal constants are limited to 18 digits on Intel, 106 on AIX and 109 on z/OS.

IBM1179I W FLOAT literal is too big for its implicit precision. An appropriate HUGE value of *assumed value* is assumed.

Explanation: The precision for a float literal is implied by the number of digits in its mantissa. For instance 1e99 is implicitly FLOAT DECIMAL(1), but the value 1e99 is larger than the largest value a FLOAT DECIMAL(1) can hold.

IBM1180I W Argument to *BUILTIN name* built-in is not byte aligned.

Explanation: This message applies to the ADDR, CURRENTSTORAGE/SIZE and STORAGE/SIZE built-in functions. Applying any one of these built-in functions to a variable that is not byte-aligned may not produce the results you expect.

IBM1181I W A WHILE or UNTIL option at the end of a series of DO specifications applies only to the last specification.

Explanation: In the following code snippet, the WHILE clause applies only to the last DO specification, that is only when I = 5;

```
do i = 1, 3, 5 while( j < 5 );
```

IBM1182I W Invocation of a NONRECURSIVE procedure from within that procedure is invalid. RECURSIVE attribute is assumed.

Explanation: A procedure contains code that will cause it to be recursively invoked, but the procedure was not declared with RECURSIVE attribute.

```
a: proc( n );
   ...
   if n > 0 then call a;
```

IBM1183I W *condition-name* condition is disabled. Statement is ignored.

Explanation: The SIGNAL statement is ignored if the condition it would raise is disabled. Some conditions, like SIZE, are disabled by default.

```
(nofofl): signal fixedoverflow;
```

IBM1184I W Source with length *string-length* in INITIAL clause for *variable name* has length greater than the length *string-length* of that INITIAL variable.

Explanation: The string in the INITIAL clause ('TooBig' in the example below) will be trimmed to fit (to 'TooB').

```
dcl x char(4) static init('tooBig');
```

IBM1185I W Source with length *string-length* in RETURN statement has length greater than that in the corresponding RETURNS attribute.

Explanation: The string in the RETURNS clause ('TooBig' in the example below) will be trimmed to fit (to 'TooB').

```
x: proc returns( char(4) );
   ...
   return( 'TooBig' );
```

IBM1186I W Source with length *string-length* in string assignment has length greater than the length *string-length* of the target.

Explanation: The source in the assignment ('TooBig' in the example below) will be trimmed to fit (to 'TooB').

```
dcl x char(4);
x = 'TooBig';
```

IBM1187I W Argument number *argument-number* in entry reference *entry name* has length *string-length* which is greater than that of the corresponding parameter.

Explanation: The source in the entry invocation

('TooBig' in the example below) will be trimmed to fit (to 'TooB').

```
dcl x entry( char(4) );
call x( 'TooBig' );
```

IBM1188I W Result of concatenating two strings is too long.

Explanation: The length of the string produced by concatenating two strings must not be greater than the maximum allowed for the derived string type.

IBM1189I W NODESCRIPTOR attribute conflicts with the NONCONNECTED attribute for the parameter *parameter name***. CONNECTED is assumed.**

Explanation: If NODESCRIPTOR is specified (or implied) for a procedure, aggregate parameters should have the CONNECTED attribute. The CONNECTED attribute can be explicitly coded, or it can be implied by the DEFAULT(CONNECTED) compiler option.

IBM1190I W The OPTIONS option *option-name* **conflicts with the LANGLVL compiler option. The option will be applied.**

Explanation: The named option is not part of the PL/I language definition as specified in the LANGLVL compiler option.

IBM1191I W Result of FIXED BIN divide will not be scaled.

Explanation: When dividing a FIXED BIN(p1,0) value by a FIXED BIN(p2,0) value where 31 > p1, the result will have the attributes FIXED BIN(p1,0). With ANSI 76, it would have the attributes FIXED BIN(31,31-p1).

IBM1192I W WHEN clauses contain duplicate values.

Explanation: In a dominated SELECT statement, if a WHEN clause has the same value as an earlier WHEN clause, the code for the second WHEN clause will never be executed. This message will be produced only if the SELECT statement is otherwise suitable for transformation into a branch table.

IBM1193I W *statement count* **statements in block** *block name***.**

Explanation: This message is produced if a block contains more statements than allowed by the MAXSTMT compiler option. It may point to blocks that are excessively large.

IBM1194I W More than one argument to MAIN procedure.

Explanation: A MAIN procedure should have at most one argument, except under SYSTEM(CICS) and SYSTEM(IMS).

IBM1195I W Argument to MAIN procedure is not CHARACTER VARYING.

Explanation: The argument to the MAIN procedure should be CHARACTER VARYING, except under SYSTEM(CICS), SYSTEM(TSO) and SYSTEM(IMS).

IBM1196I W AREA initialized with EMPTY - INITIAL attribute is ignored.

Explanation: Any INITIAL attribute specified for an AREA variable is ignored. The variable will, instead, be initialized with the EMPTY built-in function.

IBM1197I W *file-name* **assumed as file condition reference.**

Explanation: All file conditions should be qualified with a file reference, but ENDFILE and ENDPAGE are accepted without a file reference. SYSIN and SYSPRINT are then assumed, respectively.

IBM1198I W A null argument list is assumed for *variable name***.**

Explanation: An ENTRY reference is used where the result of invoking that entry is probably meant to be used.

```
dcl e1 entry returns( ptr );
dcl q  ptr based;
e1->q = null();

dcl e2 entry returns( bit(1) );
if e2 then ...
```

IBM1199I W Syntax of the %LINE directive is incorrect.

Explanation: The %LINE directive must be followed, with optional intervening blanks, by a parenthesis, a line number, a comma, a file name and a closing parenthesis.

```
%line( 19, test.pli );
```

IBM1200I W Use of DATE built-in function may cause problems.

Explanation: The DATE built-in returns a two-digit year. It might be better to use the DATETIME built-in which returns a four-digit year.

IBM1201I W *suboption* **conflicts with a previously specified suboption for the** *option* **compiler option.**

Explanation: There is a conflict of suboptions for the LANGLVL compiler option. The SAA2 and OS suboptions are mutually exclusive.

```
*process langlvl(saa2 os);
```

IBM1202I W Syntax of the %OPTION statement is incorrect.

Explanation: The only option supported in the %OPTION statement is the LANGLVL option.

IBM1203I W Argument to PLITEST built-in subroutine is ignored.

Explanation: Change the invocation of PLITEST so that no argument is passed.

IBM1204I W INTERNAL CONSTANT assumed for initialized STATIC LABEL.

Explanation: LABEL variables require block activation information, and hence they cannot be initialized at compile-time. For a STATIC LABEL variable with the INITIAL attribute, if the variable is a member of a structure or a union, a severe message will be issued. Otherwise, its attributes will be changed to INTERNAL CONSTANT in order to eliminate the requirement for block activation information. Such a variable must be initialized with LABEL CONSTANTs from containing blocks.

IBM1205I W Arguments of the *option* **compiler option must be the same length.**

Explanation: If two arguments of the NAMES option are specified, they must be the same length. The second argument is the uppercase value of the first. If a character in the first string does not have an uppercase value, use the character itself as the uppercase value. For example:

```
names( '$!@' '$!@')
```

IBM1206I W BIT operators should be applied only to BIT operands.

Explanation: In an expression of the form x & y, x | y, or x ^ y, x and y should both have BIT type.

IBM1207I W Operand to LENGTH built-in should have string type.

Explanation: If the operand has a numeric type, the result is the length that value would have after it was converted to string. The length of a numeric type is NOT the same as its storage requirement.

IBM1208I W INITIAL list for the array *variable name* **contains only one item.**

Explanation: The array will be incompletely initialized. If the named variable is part of a structure, subsequent elements in that structure with this problem will be flagged with message 2603. An asterisk can be used as an initialization factor to initialize all the elements with one value. In the example below, a(1) is initialized with the value 13, while the elements a(2) through a(8) are uninitialized. In contrast, all the elements in b are initialized to 13.

```
dcl a(8) fixed bin init( 13 );
dcl b(8) fixed bin init( (*) 13 );
```

IBM1209I W INDEXED environment option for file *file name* **will be treated as ORGANIZATION(INDEXED).**

Explanation: Since ISAM is not being simulated on the OS/2 platform, the file will be treated in a manner similar to VSAM KSDS. The file specified in the first declaration below would be handled in the same manner as the file in the second declaration. Both are treated as ORGANIZATION(INDEXED).

```
dcl f1 file env(indexed);
dcl f2 file env(organization(indexed));
```

IBM1210I W The field width specified in the *keyword***-format item may be too small for complete output of the data item.**

Explanation: The format width will be too small for output if the number is negative. It might be valid if the format is being used for input.

IBM1211I W Source with length *string-length* **in string assignment has length greater than the length** *string-length* **of the target** *variable***.**

Explanation: The source in the assignment ('TooBig' in the example below) will be trimmed to fit (to 'TooB'). If the target is a pseudovariable, message 1186 is issued instead.

```
dcl x char(4);
x = 'TooBig';
```

IBM1212I W The A format item requires an argument when used in GET statement. An L format item is assumed in its place.

Explanation: A width must be specified on A format items when specified on a GET statement.

```
get edit(name) (a);
```

IBM1213I W The procedure *proc name* is not referenced.

Explanation: The named procedure is not external and is never referenced in any live code in the compilation unit. This may represent an error (if it was supposed to be called) or an opportunity to eliminate some dead code.

IBM1214I W A dummy argument will be created for argument number *argument-number* in entry reference *entry name*.

Explanation: An argument passed BYADDR to an entry does not match the corresponding parameter in the entry description. The address of the argument will not be passed to the entry. Instead, the argument will be assigned to a temporary with attributes that do match the parameter in the entry description, and the address of that temporary will be passed to the entry. This means that if the entry alters the value of this parameter, the alteration will not be visible in the calling routine.

```
dcl e entry( fixed bin(31) );
dcl i fixed bin(15);
call e( i );
```

IBM1215I W The variable *variable name* is declared without any data attributes.

Explanation: It will be given the default attributes, but this may be because of an error in the declare. For instance, in the following example, parentheses may be missing

```
dcl a, b fixed bin;
```

IBM1216I W The structure member *variable name* is declared without any data attributes. A level number may be incorrect.

Explanation: It will be given the default attributes, but this may be because of an error in the declare. For instance, in the following example, the level number on c and d should probably be 3.

```
dcl a, b fixed bin;
  1 a,
    2 b,
      2 c,
      2 d;
```

IBM1217I W An unnamed structure member is declared without any data attributes. A level number may be incorrect.

Explanation: It will be given the default attributes, but this may be because of an error in the declare. For instance, in the following example, the level number on c and d should probably be 3.

```
dcl a, b fixed bin;
  1 a,
    2 *,
      2 c,
      2 d;
```

IBM1218I W First argument to *BUILTIN name* built-in should have string type.

Explanation: To eliminate this message, apply the CHAR or BIT built-in function to the first argument.

```
dcl i fixed bin;
display( substr(i,4) );
```

IBM1219I W LEAVE will exit noniterative DO-group.

Explanation: This message is not produced if the LEAVE statement specifies a label. In the following loop, the LEAVE statement will cause only the immediately enclosing DO-group to be exited; the loop will not be exited.

```
do i = 1 to n;
  if a(i) > 0 then
    do;
      call f;
      leave;
    end;
  else;
end;
```

IBM1220I W Result of comparison is always constant.

Explanation: This message is produced when a variable is compared to a constant equal to the largest or smallest value that the variable could assume. In the following loop, the variable x can never be greater than 99, and hence the implied comparison executed each time through the loop will always result in a '1'b.

```
dcl x pic'99';

do x = 1 to 99;
end;
```

IBM1221I W Statement uses *count* bytes for temporaries.

Explanation: This message is produced if a statement uses more bytes for temporaries than allowed by the MAXTEMP compiler option.

IBM1222I W Comparison involving 2-digit year is problematic.

Explanation: Comparisons involving data containing 2-digit year fields may cause problems if exactly one of the years is later than 1999.

IBM1223I W Literal in comparison interpreted with DATE attribute.

Explanation: In a comparison, if one comparand has the DATE attribute, the other should also. If the non-date is a literal with a value that is valid for the date pattern, it will be viewed as if it had the same DATE attribute as the date comparand. So, in the following code, '670101' will be interpreted as if it had the DATE('YYMMDD') attribute.

```
dcl x char(6) date('YYMMDD');

if x > '670101' then ...
```

IBM1224I W DATE attribute ignored in comparison with non-date literal.

Explanation: In a comparison, if one comparand has the DATE attribute, the other should also. If the non-date is a literal with a value that is not valid for the date pattern, the DATE attribute will be ignored. So, in the following code, the comparison will be evaluated as if x did not have the DATE attribute.

```
dcl x char(6) date('YYMMDD');

if x > '' then ...
```

IBM1225I W DATE attribute ignored in conversion from literal.

Explanation: If the target in an explicit or implicit assignment has the DATE attribute, the source should also. If it does not, the DATE attribute will be ignored. So, in the following code, the assignment will be performed as if x did not have the DATE attribute.

```
dcl x char(6) date('YYMMDD');

x = '';
```

IBM2600I W Compiler backend issued warning messages to STDOUT.

Explanation: Look in STDOUT to see the message issued by the compiler backend.

IBM2601I W Missing *character* assumed before *character*. DECLARE and other nonexecutable statements should not have labels.

Explanation: The indicated character is missing and has been inserted by the parser in order to correct your source.

```
xx: dcl test fixed bin;
```

IBM2602I W Number of items in INITIAL list is *count* for the array *variable name* which contains *array size* elements.

Explanation: The array will be incompletely initialized. If the named variable is part of a structure, the first element in that structure with this problem will be flagged with message 1138. This may be a programming error (in the example below, 6 should probably have been 7) and may cause exceptions when the program is run.

```
dcl
  1 a,
    2 b(8) fixed bin init( 1, (7) 29 ),
    2 c(8) fixed bin init( 1, (6) 29 );
```

IBM2603I W INITIAL list for the array *variable name* contains only one item.

Explanation: The array will be incompletely initialized. If the named variable is part of a structure, the first element in that structure with this problem will be flagged with message 1208. An asterisk can be used as an initialization factor to initialize all the elements with one value. In the example below, b(1) and c(1) are initialized with the value 13, while the elements b(2) through b(8) and c(2) through c(8) are uninitialized. In contrast, all the elements in d are initialized to 13.

```
dcl
  1 a,
    2 b(8) fixed bin init( 13 ),
    2 d(8) fixed bin init( 13 ),
    2 e(8) fixed bin init( (*) 13 );
```

IBM2604I W FIXED DEC(*source-precision*,*source-scale*) will be converted to FIXED DEC(*target-precision*,*target-scale*). Significant digits may be lost.

Explanation: If the source in a conversion to FIXED DECIMAL is a FIXED DECIMAL or PICTURE variable with a different precision and scale factor, and if the difference between the precisions is not as large as the the difference between the scale factors, then significant digits may be lost. If the SIZE condition were enabled, code would be generated to detect any such occurrence, and this message would not be issued.

```
dcl a fixed dec(04) init(1009);
dcl b fixed dec(03);

b = a;
```

IBM2605I W Invalid carriage control character. Blank assumed.

Explanation: The specified line contains an invalid ANS print control character. The valid characters are: blank, 0, -, + and 1.

IBM2606I W Code generated for the REFER object *reference name* would be more efficient if the REFER object had the attributes REAL FIXED BIN(p,0).

Explanation: If the REFER object has any other attributes, it will be converted to and from REAL FIXED BIN(31,0) via library calls.

IBM2607I W PICTURE representing FIXED DEC(*source-precision*,*source-scale*) will be converted to FIXED DEC(*target-precision*,*target-scale*). Significant digits may be lost.

Explanation: If the source in a conversion to FIXED DECIMAL is a PICTURE variable with a different precision and scale factor, and if the difference between the precisions is not as large as the the difference between the scale factors, then significant digits may be lost. If the SIZE condition were enabled, code would be generated to detect any such occurrence, and this message would not be issued.

```
dcl a pic'(4)9' init(1009);
dcl b fixed dec(03);

b = a;
```

IBM2608I W PICTURE representing FIXED DEC(*source-precision*,*source-scale*) will be converted to PICTURE representing FIXED DEC(*target-precision*,*target-scale*). Significant digits may be lost.

Explanation: If the source in a conversion to a PICTURE is a PICTURE variable with a different precision and scale factor, and if the difference between the precisions is not as large as the the difference between the scale factors, then significant digits may be lost. If the SIZE condition were enabled, code would be generated to detect any such occurrence, and this message would not be issued.

```
dcl a pic'(4)9' init(1009);
dcl b pic'(3)9';

b = a;
```

IBM2609I W Comment contains a semicolon on line *line-number*.*file-number*.

Explanation: If a comment contains a semicolon, it may indicate that there is an earlier unintentionally unclosed comment that is accidentally commenting out some source as in this example

```
/* start of unclosed comment
dcl b pic'(3)9';
/* next comment */
```

IBM2610I W One argument to *BUILTIN name* built-in is FIXED DEC while the other is FIXED BIN. Compiler will not interpret precision as FIXED DEC.

Explanation: This message applies to the MULTIPLY, DIVIDE, ADD, and SUBTRACT built-in functions: if one argument to one of these functions is FIXED DEC while the other is FIXED BIN, then the specified precision will not be interpreted as a FIXED DEC precision. This may cause improper truncation of data. For example, the result of the following multiply will have the attributes FIXED BIN(15), not FIXED DEC(15), and that might cause the result to be improperly truncated.

```
dcl a fixed bin(31);
dcl b fixed dec(15);

b = multiply( a, 1000, 15 );
```

IBM2611I W The binary value *binary value* appears in more than one WHEN clause.

Explanation: In a dominated SELECT statement, if a WHEN clause has the same value as an earlier WHEN clause, the code for the second WHEN clause will

never be executed. This message will be produced only if the SELECT statement is otherwise suitable for transformation into a branch table.

IBM2612I W The character string *character string* **appears in more than one WHEN clause.**

Explanation: In a dominated SELECT statement, if a WHEN clause has the same value as an earlier WHEN clause, the code for the second WHEN clause will never be executed. This message will be produced only if the SELECT statement is otherwise suitable for transformation into a branch table.

IBM2613I W Unless it is an output-only parameter, *variable* **may be uninitialized when used.**

Explanation: The indicated variable may be used before it has been initialized.

IBM2614I W Both comparands are Booleans.

Explanation: This message will flag statements such as the following, where the "equals" is meant to be an "and" or "or".

```
if ( a < b ) = ( c < d ) then
```

IBM2615I W DO-loop will always execute exactly once. A semicolon after the DO may be missing.

Explanation: DO-loops should normally be iterative, but if the DO-loop specification consists of just one assignment, then it will always excute once and only once. A semicolon after the DO may be missing, as in this example

```
do
  edsaup.tprs =  ads162.tprs;
  edsaup.tops =  ads162.tops;
end;
```

IBM2616I W Size of parameter *variable* **will return the currentsize value since no descriptor is available.**

Explanation: If the SIZE or STG built-in function is applied to a CHAR(*) VARYING (or VARYINGZ) parameter when there is no descriptor available, then the size of the actual storage allocated to the variable cannot be determined and only the current size can be returned.

IBM2617I W Passing a LABEL to a non-PL/I routine is very poor coding practice and will cause the compiler to generate less than optimal code.

Explanation: It is generally very unwise to pass a label to another routine. It would be good to think about redesigning any code doing this. The compiler will issue this message when a LABEL is passed to an ENTRY declared with OPTIONS(COBOL) or OPTIONS(ASM) or OPTIONS(FORTRAN). The only valid use of this label in the called routine would be to pass it on to another PL/I routine.

IBM2618I W The suboption *suboption* **is not valid for the suboption** *option* **of the** *option* **compiler option.**

Explanation: A suboption of a suboption of a compiler option is incorrect. The suboption may be unknown or outside the allowable range.

```
*process limits(extname(2000));
```

IBM2620I W Target structure contains REFER objects. Results are undefined if the assignment changes any REFER object.

Explanation: Changing REFER objects may not produce the expected results. For example, in the following example, the assignment will not change any of the elements in the array d.

```
dcl
  1 a based(p),
    2 b     fixed bin(31),
    2 c     fixed bin(31),
    2 d( 10 refer(c) ),
      3 e   fixed bin(31),
      3 f   fixed bin(31);

a = '';
```

IBM2621I W ON ERROR block does not start with ON ERROR SYSTEM. An error inside the block may lead to an infinite loop.

Explanation: The first statement in an ON ERROR block should usually be an ON ERROR SYSTEM statement. This will tend to prevent an infinite loop if there is an error in the rest of the code in the ON ERROR block.

IBM2622I W ENTRY used to set the initial value in a DO loop will be invoked after any TO or BY values are set.

Explanation: If the initial value in a DO loop is set via an ENTRY, then you may get unexpected results if that ENTRY also changes the TO or BY value. For example,

in the first loop below, the function "first" should not change the value of the variable "last". It would be better to change this code into the form of the second loop below.

```
do x = first() to last;
end;

temp = first();
do x = temp to last;
end;
```

IBM2623I W Mixing FIXED BIN and FLOAT DEC produces a FLOAT BIN result. Under DFP, this will lead to poor performance.

Explanation: Under DFP, the conversion of FLOAT DEC to FLOAT BIN requires an expensive library call that will lead to poor performance. To avoid this, the DECIMAL built-in function can be applied to the FIXED BIN operand. For example, it would be better to change the first assignment statement into the form of the second below.

```
dcl n fixed bin(31);
dcl f float dec(16);

f = n + f;
f = dec(n) + f;
```

IBM2624I W Mixing BIT and FLOAT DEC produces a FLOAT BIN result. Under DFP, this will lead to poor performance.

Explanation: Under DFP, the conversion of FLOAT DEC to FLOAT BIN requires an expensive library call that will lead to poor performance. To avoid this, the DECIMAL built-in function can be applied to the BIT operand. For example, it would be better to change the first assignment statement into the form of the second below.

```
dcl b bit(8);
dcl f float dec(16);

f = b + f;
f = dec(b) + f;
```

IBM2625I W Mixing FLOAT BIN and FLOAT DEC produces a FLOAT BIN result. Under DFP, this will lead to poor performance.

Explanation: Under DFP, the conversion of FLOAT DEC to FLOAT BIN requires an expensive library call that will lead to poor performance.

IBM2626I W Use of SUBSTR with a third argument equal to 0 is somewhat pointless since the result will always be a null string.

Explanation: While technically valid, a SUBSTR reference with a third argument that is a constant of zero probably represents a coding error.

IBM2627I W No metadata will be generated for the structure *identifier* **since its use of REFER is too complex.**

Explanation: XMI metadata is generated for BASED structures using REFER only if their use of REFER is "simple".

IBM2628I W BYVALUE parameters should ideally be no larger than 32 bytes.

Explanation: BYVALUE parameters larger than 32 bytes require too much overhead and are bad for performance.

IBM2629I W No debug symbol information will be generated for *identifier***.**

Explanation: No debug symbol information will be generated for the named variable, and hence it cannot be referenced when using the debugger.

IBM2630I W The result in an arithmetic operation has the attributes FIXED *base*(*precision*,*scale-factor*) **which means that its scale factor is greater than its precision and that the operation may lead to an overflow.**

Explanation: If the scale factor for the result of an operation exceeds the precision of the result, then unexpected fixedoverflow exceptions may occur. This can happen, for example, when multiplying two FIXED DEC(15,8) variables under the LIMITS(FIXEDDEC(15)) option because the result of such a multiplication would have the attributes FIXED DEC(15,16). To eliminate this message, the PRECISION built-in function could be used to reduce the scale factor of one of the operands or the MULTIPLY built-in function could be used to override the default attributes for the result.

IBM2631I W One argument to *BUILTIN name* **built-in is FIXED DEC while the other is FLOAT BIN. Compiler will not interpret precision as FIXED DEC.**

Explanation: This message applies to the MULTIPLY, DIVIDE, ADD, and SUBTRACT built-in functions: if one argument to one of these functions is FIXED DEC while the other is FLOAT BIN, then the specified precision will not be interpreted as a FIXED DEC precision. This may cause improper truncation of data.

For example, the result of the following multiply will have the attributes FLOAT BIN(15), not FIXED DEC(15), and that might cause the result to be improperly truncated.

```
dcl a float bin(31);
dcl b fixed dec(15);

b = multiply( a, 1000, 15 );
```

IBM2632I W One argument to *BUILTIN name* built-in is FIXED DEC while the other is FLOAT DEC. Compiler will not interpret precision as FIXED DEC.

Explanation: This message applies to the MULTIPLY, DIVIDE, ADD, and SUBTRACT built-in functions: if one argument to one of these functions is FIXED DEC while the other is FLOAT DEC, then the specified precision will not be interpreted as a FIXED DEC precision. This may cause improper truncation of data. For example, the result of the following multiply will have the attributes FLOAT DEC(15), not FIXED DEC(15), and that might cause the result to be improperly truncated.

```
dcl a float dec(15);
dcl b fixed dec(15);

b = multiply( a, 1000, 15 );
```

IBM2633I W Given the support for addressing arithmetic, basing a POINTER or OFFSET on a FIXED BIN is unnecessary, and it will also fail to work properly if the size of a POINTER changes.

Explanation: Code using such variables will work only as long as the size of the POINTER or OFFSET variable remains the same as the size of the FIXED BIN variable.

IBM2634I W Given the support for addressing arithmetic, basing a FIXED BIN on a POINTER or OFFSET is unnecessary, and it will also fail to work properly if the size of a POINTER changes.

Explanation: Code using such variables will work only as long as the size of the POINTER or OFFSET variable remains the same as the size of the FIXED BIN variable.

IBM2635I W The result in an arithmetic operation has the attributes FIXED *base*(*precision*,*scale-factor*) which means that some significant digits may be lost.

Explanation: If the scale factor for the result of an operation is negative, then the ones digits will be lost and that may cause problems. This can happen, for example, when dividing a FIXED DEC(11,2) variable by a FIXED DEC(31,29) variable because the result of such a division would have the attributes FIXED DEC(31,-7). To eliminate this message, the PRECISION built-in function could be used to reduce the scale factor of one of the operands or the DIVIDE built-in function could be used to override the default attributes for the result.

IBM2636I W The ordinal *ordinal name* appears in more than one WHEN clause.

Explanation: In a dominated SELECT statement, if a WHEN clause has the same value as an earlier WHEN clause, the code for the second WHEN clause will never be executed. This message will be produced only if the SELECT statement is otherwise suitable for transformation into a branch table.

IBM2637I W An ENTRY invoked as a function should have the RETURNS attribute.

Explanation: If an ENTRY is used as a function, it should be declared with the RETURNS attribute. The compiler will apply the RETURNS attribute to both of the ENTRYs in this example, but for E, the compiler will assume it will return FLOAT DEC while for M, it will assume it will return FIXED BIN.

```
dcl e entry;
dcl m entry;

a = e();
a = m();
```

IBM2638I W Statement used *count* intermediate language instructions.

Explanation: This message is produced if a statement uses more intermediate language instructions. than allowed by the MAXGEN compiler option. It may point to statements that are excessively complex.

IBM2639I W Previous statement used *count* intermediate language instructions.

Explanation: This message is produced if a statement uses more intermediate language instructions. than allowed by the MAXGEN compiler option. It may point to statements that are excessively complex. This message, rather than message IBM2638, is produced under the same situations as message IBM2638 except the STMT number option must also be in effect.

IBM2640I W Target is a REFER object. Results are undefined if an assignment changes a REFER object.

Explanation: Changing REFER objects might cause subsequent code to fail. For example, in the following code, the first assignment causes the second assignment to overwrite storage.

```
dcl
  1 a based(p),
    2 b     fixed bin(31),
    2 c     fixed bin(31),
    2 d( 10 refer(c) ),
      3 e   fixed bin(31),
      3 f   fixed bin(31);

allocate a;
a.c = 15;
a.f = 0;;
```

IBM2641I W The suboption *option* of the *option* compiler option must be followed by a (possibly empty) parenthesized list.

Explanation: A suboption of a compiler option has been incorrectly specified. It must be followed by a left parenthesis and then a (possibly empty) list of items and a closing right parenthesis.

```
*process deprecate(builtin);
```

IBM2642I W OPTIONS(REENTRANT) is ignored.

Explanation: Specifying OPTIONS(REENTRANT) on a PROCEDURE or BEGIN block has no effect on the generated code. Your code will be reentrant only if it does not alter any STATIC variables. You can use the DEFAULT(NONASGN) compiler option to force the compiler to flag assignments to STATIC variables.

IBM2643I W The BUILTIN function *builtin* will be deprecated.

Explanation: The named built-in function was specified in the BUILTIN suboption of the DEPRECATENEXT option, and so any explicit or contextual declaration of it is flagged.

IBM2644I W The INCLUDE file *filename* will be deprecated.

Explanation: The named INCLUDE file was specified in the INCLUDE suboption of the DEPRECATENEXT option, and so any attempt to include it is flagged.

IBM2645I W The ENTRY named *entryname* will be deprecated.

Explanation: The named ENTRY was specified in the ENTRY suboption of the DEPRECATENEXT option, and so any explicit or contextual declaration of it is flagged.

IBM2646I W The VARIABLE named *variable* will be deprecated.

Explanation: The named VARIABLE was specified in the VARIABLE suboption of the DEPRECATENEXT option, and so any explicit or contextual declaration of it is flagged.

IBM2647I W The *statementname* statement will be deprecated.

Explanation: The named statement was specified in the STMT suboption of the DEPRECATENEXT option, and so any use of that statement is flagged.

IBM2648I W Declaration contains *count* INITIAL items.

Explanation: Change the declaration to STATIC, or remove the INITIAL items and copy the INITIAL item from a STATIC variable.

IBM2649I W The binary value *binary value* appears more than once in the INLIST argument set.

Explanation: In INLIST(x, y1, y2, ...), no y value should appear twice. This message will be produced only if the INLIST function is otherwise suitable for transformation into a branch table.

IBM2650I W The ordinal *ordinal name* appears more than once in the INLIST argument set.

Explanation: In INLIST(x, y1, y2, ...), no y value should appear twice. This message will be produced only if the INLIST function is otherwise suitable for transformation into a branch table.

IBM2651I W Block *block name* contains *count* branches.

Explanation: This message is produced if a block contains more branches than allowed by the MAXBRANCH compiler option. It may point to blocks that are excessively complex.

IBM2652I W REINIT reference contains no element with an INITIAL attribute.

Explanation: In the statement REINIT x, x should contain some element with an INITIAL attribute. If not, no code will be generated for the statement.

IBM2653I W The list of preprocessor options must be enclosed in quotation marks.

Explanation: For example, rather than specifying PP(SQL(VERSION(AUTO))), specify PP(SQL('VERSION(AUTO)')).

IBM2654I W INITIAL attribute for BASED on ADDR has no effect on the base variable.

Explanation: The INITIAL attribute for BASED has an effect only if the BASED variable is used in an ALLOCATE statement. But for code such as the following, it has no effect on either the variable A or B.

```
dcl a fixed bin(31);
dcl b bit(32) based(addr(a)) init(''b);
```

IBM2655I W Some options conflict with the non-overridable options.

Explanation: If the 2 strings in the IBMZIOP module are equal, then different values for the options specified there are not allowed in the +DD options files, the invocation parameter, the options environment variable or the PROCESS statements. The conflicting options will be ignored.

IBM2656I W Simple defining applies to *variable name*. If string-overlay defining is intended, then add POS(1) to its declaration.

Explanation: In the following example, DEFBUF does not overlay the first 10 bytes of BUFFER. Instead, each array element of DEFBUF overlays the first byte of the first byte of the corresponding array element of BUFFER.

```
DCL BUFFER(10)      CHAR (300);
DCL DEFBUF(10)      CHAR(1) DEF BUFFER;
```

IBM2657I W Both logical AND operands are identical.

Explanation: This is probably a coding error.

IBM2658I W Both logical OR operands are identical.

Explanation: This is probably a coding error.

IBM2659I W Generated code would be better if all the INITIAL attributes in the declare for *variable name* were changed to VALUE.

Explanation: If an AUTOMATIC or STATIC structure consists entirely of scalar fields all of which have the INITIAL attribute and none of which have their address taken, then the compiler could probably generate much better code if all the INITIAL keywords were change to VALUE keywords. If the STATIC or AUTOMATIC attribute is explicitly specified, it would also have to be removed from the declare.

IBM2660I W Program logic may lead to the END statement for *procedure name* even though *procedure name* is a function that should return a value.

Explanation: This message warns that the compiler has detected code that could lead to an error under some conditions.

```
oops: proc( x ) returns( fixed bin(31 );
  dcl x fixed bin(31);
  select;
    when( x > 0 ) return( 1 );
    when( x = 0 ) return( 0 );
    otherwise;
  end;
end;
```

The compiler will issue this message for E15 sort exits unless the E15 sort exit specifies the OPTIONAL attribute as part of the RETURNS option on its PROCEDURE statement.

IBM2661I W The string *string value* appears more than once in the INLIST argument set.

Explanation: In INLIST(x, y1, y2, ...), no y value should appear twice. This message will be produced only if the INLIST function is otherwise suitable for transformation into a branch table.

IBM2662I W INLIST argument set contains duplicate values.

Explanation: In INLIST(x, y1, y2, ...), no y value should appear twice. This message will be produced only if the INLIST function is otherwise suitable for transformation into a branch table.

IBM2663I W WHEN clause contains an expression that matches the previous expression in the containing SELECT statement.

Explanation: In a SELECT statement, if a WHEN clause has the same expression as the previousr expression in the WHEN clauses in that SELECT statement, then the code is probably in error. The compiler will not report all such errors, but only those where an expression is duplicated in one of the four previous expressions.

IBM2664I W WHEN clause contains an expression that matches the expression *count* **previous in the containing SELECT statement.**

Explanation: In a SELECT statement, if a WHEN clause has the same expression as one of the earlier expressions in the WHEN clauses in that SELECT statement, then the code is probably in error. The compiler will not report all such errors, but only those where an expression is duplicated in one of the four previous expressions.

IBM2665I W EXTERNAL PLIXOPT declare specifies run-time options only if the variable has the attribute CHARACTER VARYING INITIAL and is not an array.

Explanation: If an EXTERNAL variable is intended to define LE runtime options, then it must be a scalar CHAR VARYING string with an INITIAL value.

IBM2666I W RETURN expression holds the address of a variable in AUTOMATIC storage.

Explanation: Returning the address of a variable in AUTOMATIC storage is likely to produce code that cannot work successfully.

IBM2667I W The string lengths in the declare for *first* **depend on the size of** *second* **whose declare comes later in the block. Consider moving the first declare after the second.**

Explanation: The extents in one declare should not depend on the size of a later declare. The compiler will swap the two declares, but this might introduce other problems. It might be better to move the first declare after the second.

IBM2668I W Using the VALUE function with the structure type *type* **adds** *count* **bytes to the generated object.**

Explanation: This message is produced if a typed structure with some VALUE attributes needs more bytes than allowed by the MAXINIT compiler option. Use of the VALUE type function will add a full copy of the structure to the generated object's constant area and may lead to binder problems.

IBM2669I W The *attribute keyword* **attribute is ignored in an ALIAS definition.**

Explanation: Attributes such as ALIGNED and UNALIGNED may be specified in a DEFINE ALIAS statement, but they will be ignored and should be removed.

Chapter 3. Compiler Error Messages (1226-1499, 2400-2599)

IBM1226I E Area extent is reduced to *maximum value*.

Explanation: The maximum size allowed for an AREA variable is 16777216.

IBM1227I E *keyword* statement is not allowed where an executable statement is required. A null statement will be inserted before the *keyword* statement.

Explanation: In certain contexts, for example after an IF-THEN clause, only executable statements are permitted. A DECLARE, DEFINE, DEFAULT or FORMAT statement has been found in one of these contexts. A null statement, (a statement consisting of only a semicolon) will be inserted before the offending statement.

IBM1228I E DEFAULT statement is not allowed where an executable statement is required. The DEFAULT statement will be enrolled in the current block, and a null statement will be inserted in its place.

Explanation: In certain contexts, for example after an IF-THEN clause, only executable statements are permitted. A DEFAULT statement has been found in one of these contexts. A null statement (a statement consisting of only a semicolon) will be inserted in place of the DEFAULT statement.

IBM1229I E FORMAT statement is not allowed where an executable statement is required. The FORMAT statement will be enrolled in the current block, and a null statement will be inserted in its place.

Explanation: In certain contexts, for example after an IF-THEN clause, only executable statements are permitted. A FORMAT statement has been found in one of these contexts. A null statement (a statement consisting of only a semicolon) will be inserted in place of the FORMAT statement.

IBM1230I E Arguments have been specified for the variable *variable name*, but it is not an entry variable.

Explanation: Argument lists are valid only for ENTRY references.

```
dcl a(15) entry returns( fixed bin(31) );
i = a(3)(4);
```

IBM1231I E Arguments/subscripts have been specified for the variable *variable name*, but it is neither an entry nor an array variable.

Explanation: Argument/subscript lists are valid only for ENTRY and array references.

```
dcl a fixed bin;
i = a(3);
```

IBM1232I E Extraneous comma at end of statement ignored.

Explanation: A comma was followed by a semicolon rather than by a valid syntactical element (such as an identifier). The comma will be ignored in order to make the semicolon valid. Under RULES(LAXPUNC), a message with the same text, but lesser severity would be issued

```
dcl 1 a, 2 b fixed bin, 2 c fixed bin, ;
```

IBM1233I E Missing *character* assumed.

Explanation: The indicated character is missing, and there are no more characters in the source. The missing character has been inserted by the parser in order to correct your source. Under RULES(LAXPUNC), a message with the same text, but lesser severity would be issued

IBM1234I E Missing *character* assumed before *character*.

Explanation: The indicated character is missing and has been inserted by the parser in order to correct your source. Under RULES(LAXPUNC), a message with the same text, but lesser severity would be issued

```
display( 'Program starting' ;
```

IBM1235I E No data format item in format list.

Explanation: Data items cannot be transmitted unless a data format item is given in the format list.

© Copyright IBM Corp. 1999, 2018

```
put edit ( (130)'-' ) ( col(1) );
```

IBM1236I E Subscripts on *keyword* labels are ignored.

Explanation: A label specified on a PROCEDURE, PACKAGE or ENTRY statement should have no subscripts.

IBM1237I E EXTERNAL ENTRY attribute is assumed for *variable-name*.

Explanation: An undeclared variable is used with an arguments list. This should give it a contextual declaration as BUILTIN, but its name is not that of a built-in function.

IBM1238I E The second argument to the *BUILTIN name* built-in is greater than the precision of the result.

Explanation: The sift amount in ISLL is should not be greater than the precision of the result.

```
i = isll( n, 221 );
```

IBM1239I E The *attribute* attribute is not supported and is ignored.

Explanation: The named attribute is either not part of the SAA PL/I language and is not supported on this platform.

```
dcl f file transient;
```

IBM1240I E The *attribute* attribute is invalid in a RETURNS descriptor.

Explanation: The RETURNS descriptor may not specify an array.

```
dcl a entry returns( (12) fixed bin );
```

IBM1241I E Only '=' and '^=' are allowed as operators in comparisons involving complex numbers.

Explanation: Equal and not equal are defined for complex variables, but you have attempted to relate them in some other way.

IBM1242I E Only '=' and '^=' are allowed as operators in comparisons involving program control data.

Explanation: Other relationships between program control data are not defined. Perhaps a variable was misspelled.

IBM1243I E REGIONAL(*integer specification (2 or 3)*) ENVIRONMENT option is not supported.

Explanation: REGIONAL(2) and REGIONAL(3) ENVIRONMENT options are syntax-checked during compile-time but are not supported during run-time.

IBM1244I E The variable specified as the *option* value in an ENVIRONMENT option must be a STATIC scalar with the attributes REAL FIXED BIN(31,0).

Explanation: This applies to the KEYLENGTH, KEYLOC and RECSIZE suboptions.

IBM1245I E The variable specified as the *option* value in an ENVIRONMENT option must be a STATIC scalar with the attribute CHARACTER.

Explanation: This applies to the PASSWORD suboption.

IBM1246I E Argument to *BUILTIN name* built-in should be CONNECTED.

Explanation: This message applies, for example, to the ADDR built-in function. The value returned by the ADDR function is the address of the first byte of its argument. If you use this pointer to refer to a based variable, the variable may be mapped over storage occupied by some other variable, rather than the storage occupied by the argument.

IBM1247I E Arithmetic operands should both be numeric.

Explanation: The required implicit conversions will be performed, but this may indicate a programming error. This message will not be issued if the RULES(LAXCONV) option is specified.

```
i = i * '2';
```

IBM1248I E Argument to *BUILTIN name* built-in should have arithmetic type.

Explanation: The argument to the named built-in function should have arithmetic type. The required implicit conversion will be performed, but this may indicate a programming error. This message will not be issued if the RULES(LAXCONV) option is specified.

IBM1249I E Argument to *BUILTIN name* built-in should have CHARACTER type.

Explanation: The argument to the named built-in function should have CHARACTER type. The required implicit conversion will be performed, but this may indicate a programming error.

IBM1252I E Argument number *argument number* to *BUILTIN name* built-in should have arithmetic type.

Explanation: The required implicit conversion will be performed, but this may indicate a programming error. This message will not be issued if the RULES(LAXCONV) option is specified.

```
x = max( x, y, z, '2' );
```

IBM1254I E Arithmetic prefix operand should be numeric.

Explanation: The required implicit conversion will be performed, but this may indicate a programming error. This message will not be issued if the RULES(LAXCONV) option is specified.

```
a = - b;
```

IBM1272I E Argument number *argument number* to *BUILTIN name* built-in is negative. It will be changed to 0.

Explanation: The second argument to built-in functions such as COPY and REPEAT must be nonnegative.

```
x = copy( y, -1 );
```

IBM1273I E Third argument to *BUILTIN name* built-in is negative. It will be changed to 0.

Explanation: The third argument to built-in functions such as COMPARE, PLIFILL, and PLIMOVE must be nonnegative.

```
call plimove( a, b, -1 );
```

IBM1274I E RULES(NOLAXIF) requires BIT(1) expressions in IF, WHILE, etc.

Explanation: Expressions in IF, WHILE, UNTIL and undominated WHEN clauses should have the attributes BIT(1) NONVARYING. If not, the expression should be compared to an appropriate null value. This message will not be issued if the RULES(LAXIF) option is specified.

```
dcl x bit(8) aligned;
 ...
if x then ...
```

IBM1281I E OPTIONS(RETCODE) on ATTACH reference is invalid and will be ignored.

Explanation: OPTIONS(RETCODE) is not supported on ATTACH references.

IBM1287I E Exponentiation operands should have numeric type.

Explanation: In an expression of the form x**y, x and y should not have string type. This message will not be issued if the RULES(LAXCONV) option is specified.

IBM1293I E WIDECHAR extent is reduced to *maximum value*.

Explanation: The maximum length allowed for a WIDECHAR variable is 32767.

IBM1294I E BIT extent is reduced to *maximum value*.

Explanation: The maximum length allowed for a BIT variable is 32767.

IBM1295I E Sole bound specified is less than 1. An upper bound of 1 is assumed.

Explanation: The default lower bound is 1, but the upper bound must be greater than the lower bound.

```
dcl x(-5) fixed bin;
```

IBM1296I E The BYADDR option conflicts with the SYSTEM option.

Explanation: The arguments passed to the MAIN procedure when SYSTEM(IMS) or SYSTEM(CICS) is in effect should not have the BYADDR attribute.

```
*process system(ims);
 a: proc( x );
   dcl x ptr byaddr;
```

IBM1297I E Source and target in BY NAME assignment have no matching assignable base identifiers.

Explanation: In a BY NAME, the source and target structures should have at least one matching base element identifier.

```
dcl 1 a, 2 b, 2 c, 2 d;
dcl 1 w, 2 x, 2 y, 2 z;
a = w, by name;
```

IBM1298I E Characters in B3 literals must be 0-7.

Explanation: In a B3 literal, each character must be either 0-7.

IBM1299I E CHARACTER extent is reduced to *maximum value***.**

Explanation: The maximum length allowed for a CHARACTER variable is 32767.

IBM1300I E *variable name* **is contextually declared as** *attribute***.**

Explanation: This is an E-level message because RULES(NOLAXDCL) has been specified.

IBM1301I E A DECIMAL exponent is required.

Explanation: An E in a FLOAT constant must be followed by at least one decimal digit (optionally preceded by a sign).

IBM1302I E The limit on the number of DEFAULT predicates in a block has already been reached. This and subsequent DEFAULT predicates in this block will be ignored.

Explanation: Each block should contain no more than 31 DEFAULT predicates.

IBM1303I E A second argument to the *BUILTIN name* **built-in must be supplied for arrays with more than one dimension. A value of 1 is assumed.**

Explanation: The LBOUND, HBOUND, and DIMENSION built-in functions require two arguments when applied to arrays having more than one dimension.

```
dcl a(5,10) fixed bin;
do i = 1 to lbound(a);
```

IBM1304I E Second argument to *BUILTIN name* **built-in is not positive. A value of 1 is assumed.**

Explanation: The DIMENSION, HBOUND and LBOUND built-in functions require that the second argument be positive.

IBM1305I E Second argument to *BUILTIN name* **built-in is greater than the number of dimensions for the first argument. A value of** *dimension count* **is assumed.**

Explanation: The second argument to the LBOUND, HBOUND, and DIMENSION built-in functions must be no greater than the number of dimensions of their array arguments.

```
dcl a(5,10) fixed bin;
do i = 1 to lbound(a,3);
```

IBM1306I E Repeated declaration of *identifier* **is invalid and will be ignored.**

Explanation: Level 1 variable names must not be repeated in the same block.

```
dcl a fixed bin, a float;
```

IBM1307I E Duplicate specification of arithmetic precision. Subsequent specification ignored.

Explanation: The precision attribute must be specified only once in a declare.

```
dcl a fixed(15) bin(31);
```

IBM1308I E Repeated declaration of *identifier* **is invalid. The name will be replaced by an asterisk.**

Explanation: The variable names at any given sublevel within a structure or union must be unique.

```
dcl 1 a, 2 b fixed, 2 b float;
```

IBM1309I E Duplicate specification of *attribute***. Subsequent specification ignored.**

Explanation: Attributes like INITIAL must not be repeated for an element of a DECLARE statement.

```
dcl a fixed init(0) bin init(2);
```

IBM1310I E The attribute *character* **conflicts with previous attributes and is ignored.**

Explanation: Attributes must be consistent.

```
dcl a fixed real float;
```

IBM1311I E EXTERNAL name contains no non-blank characters and is ignored.

Explanation: The external name should contain some nonblank characters.

```
dcl x external( '  ' );
```

IBM1312I E WX literals should contain a multiple of 4 hex digits.

Explanation: WX literals must represent unicode strings and hence must contain a multiple of 4 hex digits.

```
x = '00'wx;
```

IBM1314I E ELSE clause outside of an open IF-THEN statement is ignored.

Explanation: ELSE clauses are valid immediately after an IF-THEN statement.

```
do; if a > b then; end; else a = 0;
```

IBM1315I E END label matches a label on an open group, but that group label is subscripted.

Explanation: END statements for groups with a subscripted label must have labels that are also subscripted.

```
a(1): do;
  ...
end a;
```

IBM1316I E END label is not a label on any open group.

Explanation: A Label on END statement must match a LABEL on an open BEGIN, DO, PACKAGE, PROCEDURE, or SELECT statement.

```
a: do;
  ...
end b;
```

IBM1317I E An END statement may be missing after an OTHERWISE unit. One will be inserted.

Explanation: After an OTHERWISE unit in a SELECT statement, only an END statement is valid.

```
select;
  when ( ... )
    do;
    end;
  otherwise
    do;
    end;
display( .... );
```

IBM1318I E The ENVIRONMENT option *option-name* **conflicts with preceding ENVIRONMENT options. This option will be ignored.**

Explanation: There was a conflict detected in the ENVIRONMENT options specification. In the example ENV(CONSECUTIVE INDEXED), the INDEXED option conflicts with the CONSECUTIVE option.

IBM1319I E STRINGSIZE condition raised while evaluating expression. Result is truncated.

Explanation: During the conversion of a user expression during the compilation, the target string was found to be shorter than the source, thus causing the STRINGSIZE condition to be raised.

IBM1320I E STRINGRANGE condition raised while evaluating expression. Arguments are adjusted to fit.

Explanation: If all the arguments in a SUBSTR reference are constants or restricted expressions, the reference will be evaluated at compile- time and the STRINGRANGE condition will occur if the arguments do not comply with the rules described for the SUBSTR built-in function.

```
a = substr( 'abcdef', 5, 4 );
```

IBM1321I E LEAVE/ITERATE label matches a label on an open DO group, but that DO group label is subscripted.

Explanation: LEAVE/ITERATE statements for groups with a subscripted label must have labels that are also subscripted.

```
a(1): do;
  ...
  leave a;
```

IBM1322I E LEAVE/ITERATE label is not a label on any open DO group in its containing block.

Explanation: LEAVE/ITERATE must specify a label on an open DO loop in the same block as the LEAVE/ITERATE statement.

```
a: do loop;
  begin;
    leave a;
```

IBM1323I E ITERATE/LEAVE statement is invalid outside an open DO statement. The statement will be ignored.

Explanation: ITERATE/LEAVE statements are valid only inside DO groups.

```
a: begin;
  ...
  leave a;
  ...
end a;
```

IBM1324I E The name *name* occurs more than once in the EXPORTS clause.

Explanation: Names in the EXPORTS clause of a package statement must be unique.

```
a: package exports( a1, a2, a1 );
```

IBM1325I E The name *name* occurs in the EXPORTS clause, but is not the name of any level-1 procedure.

Explanation: Each name in the EXPORTS clause of a package statement must be the name of some level-1 procedure in that package.

```
a: package exports( a1, a2, a3 );
```

IBM1326I E Variables declared without a name must be structure members or followed by a substructure list.

Explanation: An asterisk may be used only for structure or union names, or for members of structures or unions. An asterisk may not be used for a level-1 structure name that specifies the LIKE attribute.

```
dcl * char(20) static init('who can use me');
```

IBM1327I E The CHARACTER VARYING parameter to MAIN should be ASCII with the attribute NATIVE.

Explanation: If the parameter is EBCDIC or has the attribute NONNATIVE, unpredictable results can occur.

IBM1328I E The CHARACTER VARYING parameter to MAIN should be EBCDIC with the attribute BIGENDIAN.

Explanation: If the parameter is ASCII or has the attribute LITTLEENDIAN, unpredictable results can occur. This message applies only to SYSTEM(MVS) etc.

IBM1329I E ENTRY statements are not allowed under RULES(NOMULTIENTRY).

Explanation: Under RULES(NOMULTIENTRY), there should be no ENTRY statements in your source program.

IBM1330I E The I in an iSUB token must be bigger than zero. A value of 1 is assumed.

Explanation: The I in an iSUB token must represent a valid dimension number.

```
dcl b(8) fixed bin def(0sub,1);
```

IBM1331I E The I in an iSUB token must have no more than 2 digits. A value of 1 is assumed.

Explanation: The I in an iSUB token must have only 1 or 2 digits.

```
dcl b(8) fixed bin def(001sub,1);
```

IBM1332I E The *format-item* format item requires an argument when used in GET statement. A value of 1 is assumed.

Explanation: A width must be specified on A, B, and G format items when specified on a GET statement.

```
get edit(name) (a);
```

IBM1333I E Non-asterisk array bounds are not permitted in GENERIC descriptions.

Explanation: All array bounds in generic descriptions must be asterisks.

```
dcl x generic ( e1 when( (10) fixed ), ...
```

IBM1334I E String lengths and area sizes are not permitted in GENERIC descriptions.

Explanation: All string lengths and area sizes in generic descriptions must be asterisks.

```
dcl x generic ( e1 when( char(10) ), ...
```

IBM1335I E Entry description lists are not permitted in GENERIC descriptions.

Explanation: Any ENTRY attribute in a generic description list must not be qualified with an entry description list.

```
dcl x generic ( e1 when( entry( ptr ) ), ...
```

IBM1336I E GRAPHIC extent is reduced to *maximum value***.**

Explanation: The maximum length allowed for a GRAPHIC variable is 16383.

IBM1337I E GX literals should contain a multiple of 4 hex digits.

Explanation: GX literals must represent graphic strings and hence must contain a multiple of 4 hex digits.

```
x = '00'gx;
```

IBM1338I E Upper bound is less than lower bound. Bounds will be reversed.

Explanation: A variable has been declared with an upper bound that is less than its lower bound. The upper and lower bounds will be swapped in order to correct this. For example, DECLARE x(3:1) will be changed to DECLARE x(1:3).

IBM1339I E Identifier is too long. It will be collapsed to *identifier***.**

Explanation: The maximum length of an identifier is set by the NAME suboption of the LIMITS compiler option.

IBM1340I E Argument number *argument-number* **in ENTRY reference** *ENTRY name* **contains BIT data. NOMAP is assumed.**

Explanation: An argument containing BIT data has been found in a call to a COBOL routine. Mapping of such structures between PL/I and COBOL is not supported.

```
dcl f ext entry options( cobol );

dcl 1 a, 2 b bit(8), 2 c bit(8);

call f( a );
```

IBM1341I E Argument number *argument-number* **in ENTRY reference** *ENTRY name* **is or contains a UNION. NOMAP is assumed.**

Explanation: An argument containing UNION data has been found in a call to a COBOL routine. Mapping of such structures between PL/I and COBOL is not supported.

```
dcl f ext entry options( cobol );

dcl 1 a union, 2 b char(4), 2 c fixed bin(31);

call f( a );
```

IBM1342I E Argument number *argument-number* **in ENTRY reference** *ENTRY name* **contains non-constant extents. NOMAP is assumed.**

Explanation: An argument containing non-constant extents has been found in a call to a COBOL routine. Mapping of such structures between PL/I and COBOL is not supported.

```
dcl f ext entry options( cobol );

dcl n static fixed bin init(17);

dcl 1 a, 2 b char(n), 2 c fixed bin(31);

call f( a );
```

IBM1343I E *nomap-suboption* **is invalid as a suboption of** *option***.**

Explanation: The suboption should be specified as ARGn where "n" is an integer greater than 0.

```
dcl f ext entry options( cobol nomap(arg0) );
```

IBM1344I E NOMAP specifications are valid only for ILC routines.

Explanation: NOMAP, NOMAPIN and NOMAPOUT are valid only for COBOL, FORTRAN and ASM Procedures and Entrys.

IBM1345I E Initial level number in a structure is not 1.

Explanation: The level-1 DECLARE statement may be missing.

```
dcl
  2 a,
    3 b,
    3 c,
```

IBM1346I E INIT expression should be enclosed in parentheses.

Explanation: This is required to avoid ambiguities. For example, it is unclear whether all of the elements should be initialized with the value 4 or if the first element should be initialized with the value 9.

```
dcl a(5) fixed bin init( (5)+4 );
```

IBM1347I E B assumed to complete iSUB.

Explanation: There is no language element of the form 1su.

```
dcl a(10) def b(1su, 1sub );
```

IBM1348I E Digit in BINARY constant is not zero or one.

Explanation: In a BINARY constant, each digit must be a zero or one.

IBM1349I E Characters in BIT literals must be 0 or 1.

Explanation: In a BIT literal, each character must be either zero or one.

IBM1350I E Character with decimal value *n* does not belong to the PL/I character set. It will be ignored.

Explanation: The indicated character is not part of the PL/I character set. This can occur if a program containing NOT or OR symbols is ported from another machine and those symbols are translated to a character that is not part of the PL/I character set. Using the NOT and OR compiler options can help avoid this problem.

IBM1351I E Characters in hex literals must be 0-9 or A-F.

Explanation: In a hex literal, each character must be either 0-9 or A-F.

IBM1352I E The statement element *character* is invalid. The statement will be ignored.

Explanation: The statement entered could not be parsed because the specified element is invalid.

IBM1353I E Use of underscore as initial character in an identifier accepted although invalid under LANGLVL(SAA).

Explanation: Under LANGLVL(SAA), identifiers must start with an alphabetic character or with one of the extralingual characters. They may not start with an underscore. Under LANGLVL(SAA2), identifiers may start with an underscore, although names starting with _IBM are reserved for use by IBM.

IBM1354I E Multiple argument lists are valid only with the last identifier in a reference.

Explanation: A reference of the form x(1)(2).y.z is invalid.

IBM1355I E Empty argument lists are valid only with the last identifier in a reference.

Explanation: A reference of the form x().y.z is invalid.

IBM1356I E Character with decimal value *n* does not belong to the PL/I character set. It is assumed to be an OR symbol.

Explanation: The indicated character is not part of the PL/I character set, but was immediately followed by the same character. This can occur if a program containing an OR symbol is ported from another machine and this symbol is translated to a character that is not part of the PL/I character set. Using the OR compiler option can help avoid this problem.

IBM1357I E Character with decimal value *n* does not belong to the PL/I character set. It is assumed to be a NOT symbol.

Explanation: The indicated character is not part of the PL/I character set, but was immediately followed by an =, < or > symbol. This can occur if a program containing a NOT symbol is ported from another machine and this symbol is translated to a character that is not part of the PL/I character set. Using the NOT compiler option can help avoid this problem.

IBM1358I E The scale factor specified in *BUILTIN name* built-in with a floating-point argument must be positive. It will be changed to 1.

Explanation: This applies to the ROUND built-in function. The non-positive value will be changed to 1.

```
dcl x float bin(53);
x = round( x, -1 );
```

IBM1359I E **Names in RANGE(*identifier*:*identifier*) are not in ascending order. Order is reversed.**

Explanation: The names must be in ascending order.

```
default range( h : a ) fixed bin;
```

IBM1360I E **The name *identifier* has already been defined as a FORMAT constant.**

Explanation: The name of a FORMAT constant cannot be used as the name of a LABEL constant as well.

```
f(1): format( a, x(2), a );

f(2): ;
```

IBM1361I E **The name *identifier* has already been defined as a LABEL constant.**

Explanation: The name of a LABEL constant cannot be also used as the name of a FORMAT constant.

```
f(1): ;

f(2): format( a, x(2), a );
```

IBM1362I E **The label *label-name* has already been declared. The explicit declaration of the label will not be accepted.**

Explanation: Declarations for label constant arrays are not permitted.

```
dcl a(10) label variable;

a(1): ...
a(2): ...
```

IBM1363I E **Structure level greater than 255 specified. It will be replaced by 255.**

Explanation: The maximum structure level supported is 255.

```
dcl
  1 a,
    256 b,
    2 c,
```

IBM1364I E **Elements with level numbers greater than 1 follow an element without a level number. A level number of 1 is assumed.**

Explanation: A structure level is probably missing.

```
dcl
    a,
    2 b,
    2 c,
```

IBM1365I E **Statement type resolution requires too many lexical units to be examined. The statement will be ignored.**

Explanation: To determine if a statement is an assignment or another PL/I statement, many elements of the statement may need to be examined. If too many have to be examined, the compiler will flag the statement as in error. For instance, the following statement could be a DECLARE until the equal sign is encountered by the lexer.

```
dcl ( a, b, c ) = d;
```

IBM1366I E **Level number following LIKE specification is greater than than the level number for the LIKE specification. LIKE attribute will be ignored.**

Explanation: LIKE cannot be specified on a parent structure or union.

```
dcl
  1 a like x,
    2 b,
    2 c,
```

IBM1367I E **Statements inside a SELECT must be preceded by a WHEN or an OTHERWISE clause.**

Explanation: A WHEN or OTHERWISE may be missing.

```
select;
  i = i + 1;
  when ( a > 0 )
    ...
```

IBM1368I E **The attribute *character* is invalid if it is not followed by an element with a greater logical level.**

Explanation: The named attribute is valid only on parent structures.

```
dcl
  1 a,
    2 b union,
      2 c1  fixed bin(31),
      2 c2  float bin(21),
    ...
```

IBM1369I E MAIN has already been specified in the PACKAGE.

Explanation: OPTIONS(MAIN) may be specified for only one PROCEDURE in a PACKAGE. All but the first specification will be ignored.

IBM1370I E Extent expression is negative. It will be replaced by the constant 1.

Explanation: Extents must be positive.

```
dcl x char(-10);
```

IBM1371I E Structure element *identifier* is not dot qualified.

Explanation: Under the option RULES(NOLAXQUAL), all structure elements should be qualified with the name of at least one of their parents.

IBM1372I E EXTERNAL specified on internal entry point.

Explanation: The EXTERNAL attribute is valid only on external procedures and entrys: for example, in a non-package, only on the outermost procedure and entry statements contained in it, and in a package, only on the procedures and entrys listed in the EXPORTS clause of the PACKAGE statement.

```
a: proc;
  b: proc ext('_B');
```

IBM1373I E Variable *variable name* is implicitly declared.

Explanation: Under the RULES(NOLAXDCL) option, all variables must be declared except for contextual declarations of built-in functions, SYSPRINT and SYSIN.

IBM1374I E Contextual attributes conflicting with PARAMETER will not be applied to *variable name*.

Explanation: Only those contextual attributes that can be applied to a parameter will be applied. For example, CONSTANT and EXTERNAL, which apply to contextual file declarations, will not be applied to file parameters.

```
a: proc( f );

  open file( f );
```

IBM1375I E The DEFINED variable *variable name* does not fit into its base variable.

Explanation: The number of bits, characters or graphics needed for a DEFINED variable must be no more than in the base variable.

```
dcl a char(10);

dcl b char(5) defined ( a ) pos( 8 );
```

IBM1376I E Factoring of level numbers into declaration lists containing level numbers is invalid. The level numbers in the declaration list will be ignored.

Explanation: Only attributes can be factored into declaration lists.

```
dcl 1 a, 2 ( b, 3 c, 3 d ) fixed;
```

IBM1377I E A scale factor has been specified as an argument to the *BUILTIN name* built-in, but the result of that function has type FLOAT. The scale factor will be ignored.

Explanation: Scale factors are valid only for FIXED values.

```
x = binary(1e0,4,2);
```

IBM1378I E An arguments list or subscripts list has been provided for a GENERIC entry reference. It will be ignored.

Explanation: GENERIC entry references are not allowed to contain an arguments or subscripts list.

```
dcl t generic( sub1(10) when((*)),
               sub2     when((*,*)) );
```

IBM1379I E Locator qualifier for GENERIC reference is ignored.

Explanation: GENERIC references cannot be locator-qualified.

```
dcl x generic ( ... );

call p->x;
```

IBM1380I E Target structure in assignment contains no elements with the ASSIGNABLE attribute. No assignments will be generated.

Explanation: In an assignment to a structure, some element of the structure must have the assignable attribute.

```
dcl
  1 a based,
    2 nonasgn fixed bin,
    2 nonasgn fixed bin;

p->a = 0;
```

IBM1381I E DEFINED base for a BIT structure should be aligned.

Explanation: If a BIT structure (or union) is defined on a variable that is not aligned on a byte boundary, unpredictable results may occur. This is especially true if a substructure of the DEFINED variable is passed to another routine.

IBM1382I E INITIAL attribute is invalid for STATIC FORMAT variables. Storage class is changed to AUTOMATIC.

Explanation: FORMAT variables require block activation information; they cannot be initialized at compile-time. If the variable were a member of a structure, the storage class would not be changed to AUTOMATIC, and a severe message would be issued instead.

IBM1383I E Labels on *keyword* statements are invalid and ignored.

Explanation: Labels are not permitted on DECLARE, DEFAULT, and DEFINE statements or on WHEN and OTHERWISE clauses.

IBM1384I E *message*

Explanation: This message is used to report back end error messages.

IBM1385I E Invalid DEFINED - string overlay defining attempted.

Explanation: The base variable in the DEFINED attribute must consist of UNALIGNED, NONVARYING string variables of the same string type as the DEFINED variable.

IBM1386I E DEFINED base for a BIT variable should not be subscripted.

Explanation: When one bit variable is defined on a second (the base), the base may be an array, but it must not be subscripted.

```
dcl a(20) bit(8) unaligned;
dcl b bit(8) defined( a(3) );
```

IBM1387I E The NODESCRIPTOR attribute is invalid when any parameters have * extents. The NODESCRIPTOR attribute will be ignored.

Explanation: A parameter can have * extents only if a descriptor is also passed. The NODESCRIPTOR attribute will be ignored, and descriptors will be assumed to have been passed for all array, structure and string arguments.

```
a: proc( x ) options(nodescriptor);

  dcl x char(*);
```

IBM1388I E The NODESCRIPTOR attribute is invalid when any parameters have the NONCONNECTED attribute.

Explanation: A parameter can have the NONCONNECTED attribute only if a descriptor is also passed.

```
a: proc( x ) options(nodescriptor);

  dcl x(20) fixed bin nonconnected;
```

IBM1389I E The identifier *identifier* is not the name of a built-in function. The BUILTIN attribute will be ignored.

Explanation: The BUILTIN attribute can be applied only to identifiers that are the names of built-in functions or subroutines.

IBM1390I E *note*

Explanation: This message is used by %NOTE statements with a return code of 8.

IBM1391I E **End-of-source has been encountered after an unmatched comment marker.**

Explanation: An end-of-comment marker is probably missing.

IBM1392I E **End-of-source has been encountered after an unmatched quote.**

Explanation: A closing quote is probably missing.

IBM1393I E **Item in OPTIONS list conflicts with other attributes in the declaration.** *option-name* **is ignored.**

Explanation: The indicated element of the options list is invalid.

```
dcl a file options( assembler );
```

IBM1394I E **Item in OPTIONS list is invalid for BEGIN blocks.** *option-name* **is ignored.**

Explanation: The indicated element of the options list is invalid for BEGIN blocks (although it may be valid for PROCEDUREs).

```
begin options( assembler );
```

IBM1395I E **Item in OPTIONS list is invalid for PACKAGEs.** *option-name* **is ignored.**

Explanation: The indicated element of the options list is invalid for PACKAGEs (although it may be valid for PROCEDUREs).

```
a: package exports(*) options( assembler );
```

IBM1396I E **Item in OPTIONS list is invalid for PROCEDUREs.** *option-name* **is ignored.**

Explanation: The indicated element of the options list is invalid for PROCEDUREs (although it may be valid for ENTRYs).

```
a: procedure options( inter );
```

IBM1397I E **Item in OPTIONS list is invalid for nested PROCEDUREs.** *option-name* **is ignored.**

Explanation: The indicated element of the options list is invalid for nested PROCEDUREs (although it may be valid for PROCEDUREs).

```
a: proc;
  b: proc options( main );
```

IBM1398I E **Invalid item in OPTIONS list.** *option-name* **is ignored.**

Explanation: The indicated element of the options list is not a supported option in any statement or declaration.

```
a: proc options( unknown );
```

IBM1399I E **Item in OPTIONS list is invalid for ENTRY statements.** *option-name* **is ignored.**

Explanation: The indicated element of the options list is invalid for ENTRY statements (although it may be valid for PROCEDUREs).

```
a: entry options( chargraphic );
```

IBM1400I E **Item in OPTIONS list conflicts with preceding items.** *option-name* **is ignored.**

Explanation: The elements of the options list must be consistent, unlike in the example where BYVALUE and BYADDR conflict.

```
a: proc options( byvalue byaddr );
```

IBM1401I E **Parameter attributes have been specified for a variable that is not a parameter. The parameter attributes are ignored.**

Explanation: Parameter attributes, such as BYVALUE or CONNECTED, may be specified only for parameters.

```
a: proc;
  dcl x byvalue ptr;
```

IBM1402I E **Constant in POSITION attribute is less than 1.**

Explanation: The POSITION attribute must specify a positive value.

```
dcl a def b pos(-10);
```

IBM1403I E The end of the source was reached before the logical end of the program. Null statements and END statements will be inserted as necessary to complete the program.

Explanation: The source should contain END statements for all PACKAGEs, PROCEDUREs, BEGIN blocks, DO groups, and SELECT statements, as well as statements for all IF-THEN and ELSE clauses.

IBM1404I E The procedure name *proc-name* has already been declared. The explicit declaration of the procedure name will not be accepted.

Explanation: Declarations for internal procedures are not permitted.

```
a: proc;
  dcl b entry options(byvalue);
  b: proc;
```

IBM1405I E Only one description is allowed in a returns descriptor.

Explanation: A function can return only one value.

```
dcl b entry returns( ptr, ptr );
```

IBM1406I E The product of the repetition factor *repetition-factor* and the length of the constant *string* to which it is applied is greater than the maximum length allowed for a constant. The repetition factor will be ignored.

Explanation: The string represented by a repetition factor applied to another string must conform to the same limits imposed on strings without repetition factors.

```
a = (32767) 'abc';
```

IBM1407I E Scale factor is bigger than 127. It will be replaced by 127.

Explanation: Scale factors must lie between -128 and 127 inclusive.

IBM1408I E Scale factor is less than -128. It will be replaced by -128.

Explanation: Scale factors must lie between -128 and 127 inclusive.

IBM1409I E A SELECT statement may be missing. A SELECT statement, without an expression, will be inserted.

Explanation: A WHEN or OTHERWISE clause has been found outside of a SELECT statement.

IBM1410I E Semicolon inserted after ELSE keyword.

Explanation: An END statement enclosing a statement such as DO or SELECT has been found before the statement required after ELSE.

```
do;
  if a > b then
    ...
  else
end;
```

IBM1411I E Semicolon inserted after ON clause.

Explanation: An END statement enclosing a statement such as DO or SELECT has been found before the statement required after ON condition.

```
do;
  ...
  on zdiv
end;
```

IBM1412I E Semicolon inserted after OTHERWISE keyword.

Explanation: An END statement may be misplaced or a semicolon may be missing.

IBM1413I E Semicolon inserted after THEN keyword.

Explanation: An END statement may be misplaced or a semicolon may be missing.

IBM1414I E Semicolon inserted after WHEN clause.

Explanation: An END statement may be misplaced or a semicolon may be missing.

IBM1415I E Source file does not end with the logical end of the program.

Explanation: The source file contains statements after the END statement that closed the first PACKAGE or PROCEDURE. These statements will be ignored, but their presence may indicate a programming error.

IBM1416I E Subscripts have been specified for the variable *variable name*, but it is not an array variable.

Explanation: Subscripts can be specified only for elements of an array.

IBM1417I E Second argument in SUBSTR reference is less than 1. It will be replaced by 1.

Explanation: Otherwise the STRINGRANGE condition would be raised.

IBM1418I E Second argument in SUBSTR reference is too big. It will be trimmed to fit.

Explanation: Otherwise the STRINGRANGE condition would be raised.

IBM1419I E Third argument in SUBSTR reference is less than 0. It will be replaced by 0.

Explanation: Otherwise the STRINGRANGE condition would be raised.

IBM1420I E The factor in *K/M constant* is too large and is replaced by *maximum factor*.

Explanation: The maximum K constant is 2097151K, and the maximum M constant is 2047M.

IBM1421I E More than 15 dimensions have been specified. Excess will be ignored.

Explanation: The maximum number of dimensions allowed for a variable, including all inherited dimensions, is 15.

IBM1422I E Maximum of 500 LIKE attributes per block exceeded.

Explanation: A block should contain no more than 500 LIKE references. Under LANGLVL(SAA2), there is no limit.

IBM1423I E UNALIGNED attribute conflicts with AREA attribute.

Explanation: All AREA variables must be ALIGNED.

IBM1424I E End of comment marker found when there are no open comments. Marker will be ignored.

Explanation: An */ was found when there was no open comment.

IBM1425I E There is no compiler directive *directive*. Input up to the next semicolon will be ignored.

Explanation: See the Language Reference Manual for the list of supported compiler directives.

IBM1426I E Structure level of 0 replaced by 1.

Explanation: Structure level numbers must be positive.

IBM1427I E Numeric precision of 0 replaced by 1.

Explanation: Numeric precisions must be positive.

IBM1428I E X literals should contain a multiple of 2 hex digits.

Explanation: An X literal may not contain an odd number of digits.

IBM1429I E INITIAL attribute for REFER object *variable name* is invalid.

Explanation: In DCL 1 a BASED, 2 b FIXED BIN INIT(3), 2 c(n REFER(b)), the initial clause for 'b' is invalid and may lead to unpredictable results.

IBM1430I E UNSIGNED attribute for *type type* type *type name* conflicts with negative INITIAL values and is ignored.

Explanation: If an ORDINAL type is declared with the UNSIGNED attribute, any INITIAL values specified must be nonnegative.

IBM1431I E PRECISION specified for *type type* type *type name* is too small to cover its INITIAL values and is adjusted to fit.

Explanation: An ORDINAL type must have a precision larger enough to cover the range of values defined for it.

```
define ordinal
  colors
    ( red      init(0),
      orange   init(256)
      yellow   init(512) ) unsigned prec(8);
```

IBM1432I E The type *type name* is already defined. The redefinition is ignored.

Explanation: A named type may be defined only once in any block.

IBM1433I E **The name *name* occurs more than once in the RESERVES clause.**

Explanation: Names in the RESERVES clause of a package statement must be unique.

```
a: package reserves( a1, a2, a1 );
```

IBM1434I E **The name *name* occurs in the RESERVES clause, but is not the name of any level-1 STATIC EXTERNAL variable.**

Explanation: Each name in the RESERVES clause of a package statement must be the name of some level-1 static external variable in that package.

```
a: package reserves( a1, a2, a3 );
```

IBM1435I E **A precision value less than 1 has been specified as an argument to the *BUILTIN name* built-in. It will be replaced by 15.**

Explanation: Precision values must be positive.

```
middle = divide( todo, 2, 0 );
```

IBM1436I E **The scale factor specified as an argument to the *BUILTIN name* built-in is out of the valid range. It will be replaced by the nearest valid value.**

Explanation: Scale factors must be between -128 and 127 inclusive.

```
f = fixed( i, 15, 130 );
```

IBM1437I E **The second argument to the *BUILTIN name* built-in is greater than the maximum FIXED BINARY precision. It will be replaced by the maximum value.**

Explanation: The maximum FIXED BINARY precision supported allowed depends on the FIXEDBIN suboption of the LIMITS option.

```
i = signed( n, 63 );
```

IBM1438I E **Excess arguments for ENTRY *ENTRY name* ignored.**

Explanation: More arguments were specified in an ENTRY reference than were defined as parameters in that ENTRY's declaration.

```
dcl e entry( fixed bin );
call e( 1, 2 );
```

IBM1439I E **Excess arguments for *BUILTIN name* built-in ignored.**

Explanation: More arguments were specified for the indicated built-in function than are supported by that built-in function.

```
i = acos( j, k );
```

IBM1441I E **ENTRY/RETURNS description lists for comparands do not match.**

Explanation: In a comparison of two ENTRY variables or constants, the ENTRY and RETURNS description lists should match. The linkages must also match.

```
dcl e1 entry( fixed ), e2 entry( float );

if e1 = e2 then
```

IBM1442I E **The ENTRY/RETURNS description lists in the ENTRY to be assigned to *target variable* do not match those of the target variable.**

Explanation: In an assignment of an ENTRY variable or constant, the ENTRY and RETURNS description lists for the source should match those of the target. The linkages must also match.

```
dcl e1 variable entry( fixed ), e2 entry( float );

e1 = e2;
```

IBM1443I E **An ENTRY/RETURNS description list in an ENTRY in the INITIAL list for *target variable* do not match those of the target variable.**

Explanation: When initializing an ENTRY variable or constant, the ENTRY and RETURNS description lists for the source should match those of the target. The linkages must also match.

```
dcl e1 variable entry( fixed );
dcl e2 variable entry( float ) init( e1 );
```

IBM1444I E **The ENTRY/RETURNS description lists in the RETURN statement do not match those in the corresponding RETURNS attribute**

Explanation: When a function returns an ENTRY variable or constant, the ENTRY and RETURNS description lists in the returned ENTRY reference should match those in the containing procedure's RETURNS option. The linkages must also match.

```
a: proc returns( entry( float ) );

 dcl e1 entry( fixed );

 return( e1 );
```

IBM1445I E The ENTRY/RETURNS description lists for argument number *argument-number* in entry reference *entry name* do not match those in the corresponding parameter.

Explanation: This message also occurs if the linkages do not match.

```
dcl a entry( entry( float ) );

dcl e1 entry( fixed );

call a( e1 );
```

IBM1446I E Third argument in SUBSTR reference is too big. It will be trimmed to fit.

Explanation: Otherwise the STRINGRANGE condition would be raised.

IBM1447I E Literals with an X prefix are valid only in EXEC SQL statements.

Explanation: In PL/I statements, hex literals should be specified with an X suffix.

IBM1448I E Use of nonconstant extents in BASED variables without REFER accepted although invalid under LANGLVL(SAA).

Explanation: In the SAA level-1 language definition, extents in BASED variables must all be constant except where the REFER option is used. The following would be invalid

```
dcl x based char(n);
```

IBM1449I E Use of *type function* accepted although invalid under LANGLVL(SAA).

Explanation: Type functions are not part of the SAA level-1 language.

IBM1450I E *keyword* keyword accepted although invalid under LANGLVL(SAA).

Explanation: The indicated keyword (UNSIGNED in the example below) is not defined in the SAA level-1 language.

```
dcl x fixed bin unsigned;
```

IBM1451I E Use of S, D and Q constants accepted although invalid under LANGLVL(SAA).

Explanation: The definition of the SAA level-1 language does not include S, D, and Q floating-point constants.

IBM1452I E Use of underscores in constants accepted although invalid under LANGLVL(SAA).

Explanation: The definition of the SAA level-1 language does not permit using underscores in numeric and hex constants.

IBM1453I E Use of asterisks for names in declares accepted although invalid under LANGLVL(SAA).

Explanation: The definition of the SAA level-1 language does not permit using asterisks for structure element names.

IBM1454I E Use of XN and XU constants accepted although invalid under LANGLVL(SAA).

Explanation: The definition of the SAA level-1 language does not include XN and XU constants.

IBM1455I E Use of arguments with *BUILTIN name* built-in accepted although invalid under LANGLVL(SAA).

Explanation: Under LANGLVL(SAA), the DATETIME built-in function cannot have any arguments.

```
s = datetime('DDMMYYYY');
```

IBM1456I E Use of 3 arguments with *BUILTIN name* built-in accepted although invalid under LANGLVL(SAA).

Explanation: Under LANGLVL(SAA), the VERIFY and INDEX built-in functions are supposed to have exactly 2 arguments.

```
i = verify( s, j, k );
```

IBM1457I E Use of 1 argument with *BUILTIN name* built-in accepted although invalid under LANGLVL(SAA).

Explanation: Under LANGLVL(SAA), the DIM, LBOUND and HBOUND built-in functions are supposed to have 2 arguments.

```
i = dim( a );
```

IBM1458I E GOTO is not allowed under RULES(NOGOTO).

Explanation: Under RULES(NOGOTO(STRICT)), there should be no GOTO statements in your source program except for those that exit an ON-unit.

IBM1459I E Uninitialized AUTOMATIC variables in a block should not be used in the prologue of that block.

Explanation: The AUTOMATIC variables in a block may be used in the declare statements and the executable statements of any contained block, but in the block in which they are declared, they should be used only in the executable statements.

```
dcl x fixed bin(15) automatic;
dcl y(x) fixed bin(15) automatic;
```

IBM1460I E Under RULES(ANS), nonzero scale factors are not permitted in declarations of FIXED BIN. Declared scale factor will be ignored.

Explanation: RULES(IBM) allows scaled FIXED BIN, but RULES(ANS) supports it only for FIXED DECIMAL. RULES(ANS) will ignore the scale factors in the following declares

```
dcl x fixed bin(31,16);
dcl y entry( fixed bin(31,16) );
```

IBM1461I E Under RULES(ANS), nonzero scale factors are not permitted when the result of *BUILTIN name* has the attributes FIXED BIN. Specified scale factor will be ignored.

Explanation: RULES(IBM) allows scaled FIXED BIN, but RULES(ANS) supports it only for FIXED DECIMAL. RULES(ANS) will ignore the scale factors in the following built-ins

```
dcl (x,y) fixed bin(15,0);
put list( add(x,y,31,2) );
put list( bin(x,31,2) );
put list( prec(x,31,2) );
```

IBM1462I E Expression in comparison interpreted with DATE attribute.

Explanation: In a comparison, if one comparand has the DATE attribute, the other should also. If the non-date is an expression that could have a value that is valid for the date pattern, it will be viewed as if it had the same DATE attribute as the date comparand.

IBM1463I E Operand with DATE attribute is invalid except in compare or assign. DATE attribute will be ignored.

Explanation: Comparisons are the only infix operations where operands with the DATE attribute may be used. If they are used in any other operation, the DATE attribute will be ignored. So, in the following code, the addition will be flagged and the DATE attribute ignored.

```
dcl x char(5) date('YYDDD');

put list( x + 1 );
```

IBM1464I E DATE attribute ignored in comparison with non-date expression.

Explanation: In a comparison, if one comparand has the DATE attribute, the other should also. If the non-date is an expression that could not have a value that is not valid for the date pattern, the DATE attribute will be ignored.

IBM1465I E Source in assignment has the DATE attribute, but target *variable* does not. The DATE attribute will be ignored.

Explanation: If the target in an assignment has the DATE attribute, the source should also. If the target is a pseudovariable, message 1466 is issued instead.

```
dcl x char(6);
x = date();
```

IBM1466I E Source in assignment has the DATE attribute, but target does not. The DATE attribute will be ignored.

Explanation: If the source in an assignment has the DATE attribute, the target should also.

IBM1467I E **Source in INITIAL clause for *variable name* has the DATE attribute but the target does not. The DATE attribute will be ignored.**

Explanation: If an INITIAL expression has the DATE attribute, the target should also.

IBM1468I E **Argument number *argument-number* in entry reference *entry name* has the DATE attribute but the corresponding parameter does not. The DATE attribute will be ignored.**

Explanation: The argument and parameter should match, unlike in the example below

```
dcl x entry( char(6) );
call x( date() );
```

IBM1469I E **Source in RETURN statement has the DATE attribute, but the corresponding RETURNS option does not. The DATE attribute will be ignored.**

Explanation: The attributes of the RETURNed expression and in the RETURNS option should match, unlike in the example below

```
x: proc returns( char(6) );
   ...
   return( date() );
```

IBM1470I E **An ID option must be specified for the INCLUDE preprocessor.**

Explanation: No other options are valid for the INCLUDE preprocessor.

IBM1471I E **The ID option specified for the INCLUDE preprocessor is invalid.**

Explanation: The INCLUDE preprocessor ID option must have one suboption consisting of a string specifying the INCLUDE directive.

IBM1472I E **A closing right parenthesis is missing from the ID option specified for the INCLUDE preprocessor.**

Explanation: The suboption specified for the INCLUDE preprocessor ID option must be closed with a right parenthesis.

IBM1473I E **The syntax of the preprocessor INCLUDE directive is incorrect.**

Explanation: A statement that starts with the preprocessor INCLUDE directive specified in that preprocessor's ID option must be followed by a name and, optionally, a semicolon.

IBM1474I E **Source in assignment does not have the DATE attribute, but target *variable* does. The DATE attribute will be ignored.**

Explanation: If the target in an assignment has the DATE attribute, the source should also. If the target is a pseudovariable, message 1475 is issued instead.

```
dcl x char(6) date('YYMMDD');
x = '';
```

IBM1475I E **Target in assignment has the DATE attribute, but source does not. The DATE attribute will be ignored.**

Explanation: If the target in an assignment has the DATE attribute, the source should also.

IBM1476I E **Source in INITIAL clause for *variable name* does not have the DATE attribute but the target does. The DATE attribute will be ignored.**

Explanation: If a variable has the DATE attribute, then any INITIAL value for it should also.

IBM1477I E **Argument number *argument-number* in entry reference *entry name* does not have the DATE attribute but the corresponding parameter does. The DATE attribute will be ignored.**

Explanation: The argument and parameter should match, unlike in the example below

```
dcl x entry( char(6) date('YYMMDD') );
call x( '' );
```

IBM1478I E **Source in RETURN statement does not have the DATE attribute, but the corresponding RETURNS option does. The DATE attribute will be ignored.**

Explanation: The attributes of the RETURNed expression and in the RETURNS option should match, unlike in the example below

```
x: proc returns( char(6) date('YYMMDD') );
   ...
   return( '' );
```

IBM1479I E Multiple RETURN statements are not allowed under RULES(NOMULTIEXIT).

Explanation: Under RULES(NOMULTIEEXIT), there should be at most one RETURN statement in each PROCEDURE and BEGIN block in your source program.

IBM1480I E Multiple closure of groups is not allowed under RULES(NOMULTICLOSE).

Explanation: Under RULES(NOMULTICLOSE), there should be no multiple closure of groups in your source program.

IBM1481I E BYNAME assignment statements are not allowed under RULES(NOBYNAME).

Explanation: Under RULES(NOBYNAME), there should be no BYNAME assignment statements in your source program.

IBM1482I E The variable *variable name* is declared without any data attributes.

Explanation: It will be given the default attributes, but this may be because of an error in the declare. For instance, in the following example, parentheses may be missing. Under RULES(LAXDCL), this is a W-level message.

```
dcl a, b fixed bin;
```

IBM1483I E The structure member *variable name* is declared without any data attributes. A level number may be incorrect.

Explanation: It will be given the default attributes, but this may be because of an error in the declare. For instance, in the following example, the level number on c and d should probably be 3. Under RULES(LAXDCL), this is a W-level message.

```
dcl a, b fixed bin;
  1 a,
    2 b,
      2 c,
      2 d;
```

IBM1484I E An unnamed structure member is declared without any data attributes. A level number may be incorrect.

Explanation: It will be given the default attributes, but this may be because of an error in the declare. For instance, in the following example, the level number on c and d should probably be 3. Under RULES(LAXDCL), this is a W-level message.

```
dcl a, b fixed bin;
  1 a,
    2 *,
      2 c,
      2 d;
```

IBM2400I E Compiler backend issued error messages to STDOUT.

Explanation: Look in STDOUT to see the message issued by the compiler backend.

IBM2401I E Missing *character* assumed before *character*. DECLARE and other nonexecutable statements should not have labels.

Explanation: The indicated character is missing and has been inserted by the parser in order to correct your source. Under RULES(LAXPUNC), a message with the same text, but lesser severity would be issued

```
xx: dcl test fixed bin;
```

IBM2402I E *variable name* is declared as BASED on the ADDR of *variable name*, but *variable name* requires more storage than *variable name*.

Explanation: The amount of storage needed for a BASED variable must be no more than provided by its base variable.

```
dcl a char(10);

dcl b char(15) based(addr(a));
```

IBM2403I E PROCESS statements are not permitted under the NOPROCESS option.

Explanation: When the NOPROCESS option is in effect, the source should contain no PROCESS statements.

IBM2404I E *variable name* is declared as BASED on the ADDR of *variable name*, but *variable name* requires more storage than remains in the enclosing level 1 structure *variable name* after the location of *variable name*.

Explanation: The amount of storage needed for a BASED variable must be no more than provided by its base variable.

```
dcl 1 a, 2 a1 char(10), 2 a2 char(10);

dcl b char(15) based(addr(a2));
```

IBM2405I E Even decimal precisions are not allowed under RULES(NOEVENDEC).

Explanation: Under RULES(NOEVENDEC), there should be no FIXED DECIMAL data declared with an even precision.

```
dcl a fixed dec(10);
```

IBM2406I E Precision outside VALUE clause will be ignored.

Explanation: In DEFAULT statements, numeric precisions should be specified only inside VALUE clauses.

```
dft range(*) fixed bin(31);
```

IBM2407I E Length outside VALUE clause will be ignored.

Explanation: In DEFAULT statements, lengths of strings should be specified only inside VALUE clauses.

```
dft range(*) bit(8);
```

IBM2408I E AREA size outside VALUE clause will be ignored.

Explanation: In DEFAULT statements, sizes of AREAs should be specified only inside VALUE clauses.

```
dft range(*) area(10000);
```

IBM2409I E RETURN statement without an expression is invalid inside a subprocedure that specified the RETURNS attribute.

Explanation: All RETURN statements inside functions must specify a value to be returned.

```
a: proc returns( fixed bin );

  return;
```

IBM2410I E Function *function name* contains no valid RETURN statement.

Explanation: Functions must contain at least one RETURN statement.

IBM2411I E STRINGOFGRAPHIC(CHARACTER) option is ignored because argument to STRING built-in function is possibly not contiguous.

Explanation: The STRINGOFGRAPHIC(CHARACTER) option will be ignored if the argument contains any elements that are VARYING or if the argument is a NONCONNECTED slice of an array.

IBM2412I E Procedure has no RETURNS attribute, but contains a RETURN statement. A RETURNS attribute will be assumed.

Explanation: If a procedure contains a RETURN statement, it should have the RETURNS attribute specified on its PROCEDURE statement.

```
a: proc;
  return( 0 );
end;
```

IBM2413I E The attribute *attribute* should be specified only on parameters and descriptors.

Explanation: Attributes must be consistent.

```
dcl a fixed based connected;
```

IBM2414I E The *option* option conflicts with the *option* option. The IBM default of *option* will be used instead.

Explanation: The specified options conflict and cannot be used together. On ASCII systems, the compiler will produce this message if you specify the GRAPHIC and EBCDIC options. Conversely, on EBCDIC systems, the compiler will produce this message if you specify the GRAPHIC and ASCII options.

IBM2415I E Without APAR *number*, compiler would generate incorrect code for this statement.

Explanation: The indicated APAR will fix a compiler problem with this statement.

IBM2416I E The SEPARATE suboption of TEST is not supported when the LINEDIR option is in effect.

Explanation: When the LINEDIR option is in effect, only the NOSEPARATE suboption of the TEST option is supported.

IBM2417I E In FETCHABLE code compiled with NORENT NOWRITABLE(PRV), it is invalid to ALLOCATE or FREE a CONTROLLED variable unless it is a PARAMETER.

Explanation: In FETCHABLE code, all CONTROLLED variables should be parameters.

IBM2418I E Variable *variable* is unreferenced.

Explanation: The compiler will issue this message for any level-1 variable that is not referenced in a particular storage class named in the RULES option: for example, AUTOMATIC variables under RULES(NOUNREF), BASED variables under RULES(NOUNREFBASED), etc

IBM2419I E *option* is invalid and ignored unless the ARCH option is *level* or greater.

Explanation: The RTCHECK option will be ignored unless the ARCH option is 8 or greater since the necessary instructions are available only with ARCH(8) or later.

IBM2420I E DFP is invalid and ignored unless the ARCH option is 7 or greater.

Explanation: The FLOAT(DFP) option will be ignored unless the ARCH option is 7 or greater since the necessary instructions are available only with ARCH(7) or later.

IBM2421I E A file should not be closed in its ENDFILE block.

Explanation: In an ENDFILE block for a file, it is invalid to close that file in the ENDFILE block.

IBM2422I E Under the DFP option, the HEXADEC attribute is not supported for FLOAT DEC.

Explanation: Under the FLOAT(DFP) option, all FLOAT DECIMAL will be treated as DFP and may not be declared as HEXADEC. The attribute is still valid for FLOAT BIN.

IBM2423I E Under the DFP option, the IEEE attribute is not supported for FLOAT DEC.

Explanation: Under the FLOAT(DFP) option, all FLOAT DECIMAL will be treated as DFP and may not be declared as IEEE. The attribute is still valid for FLOAT BIN.

IBM2424I E Scale factors are not allowed in FLOAT declarations.

Explanation: Scale factors are valid only in declares of FIXED BIN or FiXED DEC. The first declaration below is invalid and should be changed to one of the subsequent declarations.

```
dcl a1 float dec(15,2);

dcl a2 fixed dec(15,2);
dcl a3 float dec(15);
```

IBM2425I E Statement with ELSE IF should be rewritten using SELECT.

Explanation: Under RULES(NOELSEIF), the compiler will issue this message for statement where an ELSE is immediately followed by an IF statement.

IBM2426I E Maximum nesting of DO statements has been exceeded.

Explanation: The nesting of DO statements has exceeded the value specified in the DO suboption of the MAXNEST compiler option.

IBM2427I E Maximum nesting of IF statements has been exceeded.

Explanation: The nesting of IF statements has exceeded the value specified in the IF suboption of the MAXNEST compiler option.

IBM2428I E Maximum nesting of PROC and BEGIN statements has been exceeded.

Explanation: The nesting of PROC and BEGIN statements has exceeded the value specified in the BLOCK suboption of the MAXNEST compiler option.

IBM2429I E CMPAT(V3) requires that 8-byte integers be allowed. The second value in the FIXEDBIN suboption of the LIMITS option will be set to 63.

Explanation: The use of the CMPAT(V3) option with LIMITS(FIXEDBIN(31,31)) is not supported. Since CMPAT(V3) will cause various built-in functions (such as HBOUND) to return a FIXED BIN(63) result, at least the second value in the FIXEDBIN suboption of LIMITS must be 63 (i.e. LIMITS(FIXEDBIN(31,63)) or LIMITS(FIXEDBIN(63,63)) must be in effect).

IBM2430I E The LINESIZE value specified in the OPEN of file *file name* is not compatible with the RECSIZE specified in its declare.

Explanation: If the file has F format and is not a

PRINT file, then the LINESIZE must be no greater than the RECSIZE. If the file has F format and is a PRINT file, then the LINESIZE must be less than the RECSIZE. If the file has V format and is not a PRINT file, then the LINESIZE must be no greater than the RECSIZE-4. If the file has V format and is a PRINT file, then the LINESIZE must be less than the RECSIZE-4.

IBM2431I E The *option* option conflicts with the GOFF option. NOGOFF will be used instead.

Explanation: The specified option is not permitted with the GOFF option, and the GOFF option will be turned off so that the compile may proceed. This applies, for example, to the NOWRITABLE(PRV) and COMMON options.

IBM2432I E The attribute *character* is invalid with parameters and is ignored.

Explanation: The INITIAL attribute, for example, is invalid with parameters (since their storage will have been allocated elsewhere).

```
dcl a fixed bin parameter initial( 0 );
```

IBM2433I E The attribute *character* is invalid with DEFINED and is ignored.

Explanation: The INITIAL attribute, for example, is invalid with DEFINED variables (since their storage will have been allocated elsewhere).

```
dcl b char(1) initial( '' ) defined(a);
```

IBM2434I E Under RULES(NOLAXENTRY), all ENTRY declares must specify a parenthesized parameter list, even if empty.

Explanation: Under RULES(NOLAXENTRY), all ENTRY declares must be prototyped. If the ENTRY should have no parameters, it should be declared as ENTRY() rather than as simply ENTRY.

IBM2435I E Scale factor is less than 0.

Explanation: Under RULES(NOLAXSCALE), scale factors must be nonnegative, and the compiler flags the statement below.

```
dcl a fixed dec(15,-2);
```

IBM2436I E Scale factor is larger than the precision.

Explanation: Under RULES(NOLAXSCALE), scale factors must be no larger than the precision,

```
dcl a fixed dec(15,17);
```

IBM2437I E SQL preprocessor invoked more than once without INCONLY.

Explanation: If the SQL preprocessor is invoked more than once without INCONLY as its suboption, then the DBRM library member created for the compile will be empty. It is best to invoke the SQL preprocessor either only once or once with INCONLY as its only suboption and then only once more.

IBM2438I E STOP and EXIT statements are not allowed.

Explanation: Under RULES(NOSTOP), there should be no STOP and no EXIT statements in your source program.

IBM2439I E END statement for a PROCEDURE must include the name of the PROCEDURE.

Explanation: Under RULES(NOPROCENDONLY), the END statement for a PROCEDURE must not consist of simply the END keyword and a semicolon. It must also include the name of the PROCEDURE it is closing.

IBM2440I E Structure element *identifier* is not qualified with the name of its containing level-1 structure.

Explanation: Under the option RULES(NOLAXQUAL), all structure elements should be qualified with the name of their outermost parent.

IBM2441I E GOTO exits the current block.

Explanation: Under RULES(NOGOTO(LOOSE)) and RULES(NOGOTO(LOOSEFORWARD)), there should be no GOTO statements in your source program except for those that exit an ON-unit and those that goto a label in the current block.

IBM2442I E Structure *identifier* contains padding.

Explanation: Under RULES(NOPADDING), structures should contain no padding.

IBM2443I E Control variable in DO statement belongs to a parent block.

Explanation: Under RULES(NOGLOBALDO), in a DO loop of the form DO x = .., x must be declared in the same block as the DO loop.

IBM2444I E The BUILTIN function *builtin* has been deprecated.

Explanation: The named built-in function was specified in the BUILTIN suboption of the DEPRECATE option, and so any explicit or contextual declaration of it is flagged.

IBM2445I E The INCLUDE file *filename* has been deprecated.

Explanation: The named INCLUDE file was specified in the INCLUDE suboption of the DEPRECATE option, and so any attempt to include it is flagged.

IBM2446I E The ENTRY named *variable* has been deprecated.

Explanation: The named ENTRY was specified in the ENTRY suboption of the DEPRECATE option, and so any explicit or contextual declaration of it is flagged.

IBM2447I E The VARIABLE named *variable* has been deprecated.

Explanation: The named VARIABLE was specified in the VARIABLE suboption of the DEPRECATE option, and so any explicit or contextual declaration of it is flagged.

IBM2448I E CICS preprocessor invoked more than once.

Explanation: If the CICS preprocessor were invoked more than once, then the second invocation would cause duplicate declarations to be inserted in the outermost procedure. The CICS preprocessor must be invoked only once. The compiler ignores any excess invocations.

IBM2449I E Source and target in assignment are identical.

Explanation: Under RULES(NOSELFASSIGN), the source and target in an assignment must be different.

IBM2450I E First argument to *BUILTIN name* built-in should have length greater than or equal to *length*.

Explanation: The argument to the named built-in function is too short. For example, the argument to the Y4DATE built-in function should have the form YYMMDD with possibly some trailing blanks, and hence the length of that argument should be greater than or equal to 6.

IBM2451I E Source in the assignment is a Boolean, but the target is not BIT(1).

Explanation: Under RULES(NOLAXIF), if the target in an assignment is not BIT(1), the assignment is flagged if the source is a Boolean. So, for example, the first assignment below is correct, but RULES(NOLAXIF) flags the second assignment since the third assignment might be what was intended.

```
x = (y = z);

x = y = z;

x, y = z;
```

IBM2452I E Scale factor is less than 0.

Explanation: Under RULES(NOLAXSCALE), scale factors must be nonnegative. The compiler flags the first statement below, but not the second one (which is a possible replacement for the first).

```
b = round( c, -1 );

b = 10 * round( c/ 10, 0 );
```

IBM2453I E Code should not come after a nested procedure.

Explanation: Under RULES(NOLAXNESTED), all executable code in a procedure must come before its first nested subprocedure.

IBM2454I E The *builtin* statement has been deprecated.

Explanation: The named statement was specified in the STMT suboption of the DEPRECATE option, and so any use of that statement is flagged.

IBM2455I E The *builtin* keyword does not conform to the CASERULES option.

Explanation: The named keyword does not follow the case rules specified in the KEYWORD suboption of the CASERULES option.

IBM2456I E RECURSIVE procedures are not allowed under RULES(NORECURSIVE).

Explanation: Under RULES(NORECURSIVE), the RECURSIVE attribute should not be used and procedures should not call themselves.

IBM2457I E RULES(NORECURSIVE) conflicts with DFT(RECURSIVE). The compiler will apply RULES(RECURSIVE) instead.

Explanation: If you want to use DFT(RECURSIVE), then RULES(RECURSIVE) should also be used. If RULES(NORECURSIVE) is more important, then DFT(NONRECURSIVE) should be used.

IBM2458I E The CONTROLLED attribute is not allowed under RULES(NOCONTROLLED).

Explanation: Under RULES(NOCONTROLLED), the CONTROLLED attribute must not be used.

IBM2459I E The characters specified in the *option* option must all have hexadecimal values less than '80'x.

Explanation: Under the ENCODING(UTF8) option, the characters specified in the OR, NOT, QUOTE, and BLANK compiler options must all be one-byte UTF-8 characters.

IBM2460I E The *option* option conflicts with the ENCODING(UTF8) option. ENCODING(ASCII) will be assumed.

Explanation: The specified options conflict and cannot be used together. The ENCODING(UTF8) option cannot be used with the SOSI, DBCS or GRAPHIC options.

IBM2461I E The MARGINI option must specify a valid UTF-8 string consisting of one UTF-8 character.

Explanation: Under the ENCODING(UTF8) option, the MARGINI option must be a one-character UTF-8 string. If not, a blank will be used instead.

IBM2462I E The attribute *character* conflicts with the attribute *character* and is ignored.

Explanation: Attributes must be consistent.

```
dcl a parameter static;
```

IBM2463I E LINKAGE(SYSTEM) is not supported for PL/I procedures, and LINKAGE(OPTLINK) will be assumed instead.

Explanation: Under 64-bit, only the OPTLINK linkage is supported for PL/I procedures

IBM2464I E Line contains more than one statement.

Explanation: Under RULES(NOLAXSTMT), there should be only one statement per line.

IBM2465I E Assignment of a null string to a pointer is invalid.

Explanation: Under DEFAULT(NULLSTRPTR(STRICT)), such assignments are invalid.

IBM2466I E Comparison of a null string to a pointer is invalid.

Explanation: Under DEFAULT(NULLSTRPTR(STRICT)), such comparisons are invalid.

IBM2467I E RULES(NOYY) conflicts with use of a date pattern with a 2-digit year.

Explanation: Under RULES(NOYY), the use of date patterns with a 2-digit year is invalid.

IBM2468I E RULES(NOYY) conflicts with use of a date pattern with a ZY.

Explanation: Under RULES(NOYY), the use of date patterns with a ZY is invalid.

IBM2469I E RULES(NOYY) conflicts with use of the DATE attribute without a pattern.

Explanation: Under RULES(NOYY), the use of the DATE attribute without a pattern is invalid since it implies a pattern of YYMMDD.

IBM2470I E RULES(NOYY) conflicts with use of the *BUILTIN name* built-in function.

Explanation: Under RULES(NOYY), the use of any of the Y4 date built-in functions is invalid.

IBM2471I E RULES(NOYY) conflicts with use of the *BUILTIN name* built-in function with a window argument.

Explanation: Under RULES(NOYY), the use of any date built-in function with a window argument is invalid.

IBM2472I E RULES(NOYY) conflicts with use of the DATE built-in function.

Explanation: Under RULES(NOYY), the use of the DATE built-in functions is invalid since it will return a 2-digit year.

IBM2473I E *proc name* **has not been explicitly declared.**

Explanation: Under RULES(NOLAXINTERFACE), if there is a PACKAGE statement, then every external PROCEDURE other than MAIN must be declared.

IBM2474I E GOTO jumps to a previous line in the current block.

Explanation: Under RULES(NOGOTO(LOOSEFORWARD)), there should be no GOTO statements in your source program except for those that exit an ON-unit and those that goto a label on a later line in the current block.

IBM2475I E Line contains too many semicolons.

Explanation: Under RULES(NOMULTISEMI), there should be only one semicolon on a line.

IBM2476I E Item in OPTIONS list is invalid for ON-unit BEGIN blocks. *option-name* **is ignored.**

Explanation: The indicated element of the options list is invalid for ON-unit BEGIN blocks (although it may be valid for other BEGIN blocks).

```
on zdiv begin options( inline );
```

IBM2477I E Variable *variable* **is used, but not set.**

Explanation: The compiler will issue this message for any level-1 automatic variable that is used, but not the target of an assignment statement if the RULES(NOUNSET) option is in effect.

IBM2478I E Under RULES(NOCOMPLEX), the COMPLEX attribute, the COMPLEX built-in function, and constants ending with the I suffix are not allowed.

Explanation: Under RULES(NOCOMPLEX), the COMPLEX attributes, the COMPLEX built-in function, and "imaginary" constants (such as 1i) must not be used.

IBM2479I E Compilation unit does not contain a PACKAGE statement.

Explanation: Under RULES(NOLAXPACKAGE), every compilation unti must contain a PACKAGE statement.

IBM2480I E Package contains procedures but no EXPORTS clause naming specifically which procedures are exported.

Explanation: Under RULES(NOLAXEXPORTS), every PACKAGE that contains procedures must have an EXPORTS clause that names the routines it exports.

IBM2481I E Scale factor is greater than 0.

Explanation: Under RULES(NOLAXSCALE(STRICT)), scale factors for FIXED BIN must be zero. The compiler uses other messages to flag negative scale factors and scale factors greater than the precision, but it uses this message to flag all other positive scale factors such as in the statement below.

```
dcl a fixed bin(15,2);
```

IBM2482I E Parameter *variable* **is declared without INONLY, OUTONLY, or INOUT.**

Explanation: If the RULES(NOLAXPARMS) option is in effect, The compiler will issue this message for any level-1 paramter declared without specifying if it is an input, an output or both.

IBM2483I E The structure *identifier* **is** *count***-byte aligned, but does not have a multiple of** *count* **bytes before its first element with that alignment.**

Explanation: Under RULES(NOPADDING(STRICT)), structures should contain no hang.

IBM2484I E The structure *identifier* **does not have a multiple of 8 bits before its first element with byte (or greater) alignment.**

Explanation: Under RULES(NOPADDING(STRICT)), structures should contain no hang.

IBM2485I E The size of the structure *identifier* **is not a multiple of its alignment.**

Explanation: Under RULES(NOPADDING(STRICT)), structures should contain no padding.

IBM2486I E The structure *identifier* **does not have a multiple of 8 bits after its last element with byte (or greater) alignment.**

Explanation: Under RULES(NOPADDING(STRICT)), structures should contain no hang.

IBM2487I E The structure *identifier* **does not contain a multiple of 8 bits.**

Explanation: Under RULES(NOPADDING(STRICT)), structures should contain no hang.

IBM2489I E **FIXED DEC(*source-precision*,*source-scale*) operand will be converted to FIXED BIN(*target-precision*,*target-scale*). This introduces a non-zero scale factor into an integer operation and will produce a result with the attributes FIXED BIN(*result-precision*,*result-scale*).**

Explanation: Under RULES(IBM), when an arithmetic operation has an operand that is FIXED BIN and an operand that is FIXED DEC with a non-zero scale factor, then the FIXED DEC operand will be converted to FIXED BIN. Under RULES(NOLAXSCALE(STRICT)), this is flagged as an error.

IBM2490I E **Source in assignment does not fit in the the VALUERANGE of the target.**

Explanation: When assigning to a target with the VALUERANGE attribute, the source must have a value in that range.

IBM2491I E **Source in assignment does not occur in the the VALUELIST of the target.**

Explanation: When assigning to a target with the VALUELIST attribute, the source must have a value in that list.

IBM2492I E **RULES(NOGLOBAL) violation: Variable *variable* is used inside a nested PROCEDURE.**

Explanation: If the RULES(NOGLOBAL) option is in effect, the compiler will issue this message for variables that are used in a procedure that is nested inside the procedure in which they were declared.

Chapter 4. Compiler Severe Messages (1500-2399)

IBM1500I S **Argument number** *argument-number* **in ENTRY reference** *ENTRY name* **has type** *source type*, **which is invalid for a parameter with type** *target type*.

Explanation: An argument must have a type that can be converted to the corresponding parameter's type.

IBM1501I S **Argument number** *argument-number* **in ENTRY reference** *ENTRY name* **has a different strong type than the corresponding parameter.**

Explanation: If a parameter is strongly typed, any argument passed to it must have the same type.

IBM1502I S **Argument number** *argument-number* **in ENTRY reference** *ENTRY name* **has type** *source type*, **which is invalid for a parameter with type** *target type*. **If the ENTRY should be invoked, an argument list must be provided.**

Explanation: An argument must have a type that can be converted to the corresponding parameter's type.

IBM1503I S **Argument number** *argument-number* **in ENTRY reference** *ENTRY name* **has type** *source type*, **which is invalid for a parameter with type LIMITED ENTRY.**

Explanation: Only an EXTERNAL ENTRY CONSTANT, an ENTRY CONSTANT representing a non-nested PROCEDURE, or an ENTRY VARIABLE with the LIMITED attribute can be passed to a LIMITED ENTRY parameter.

IBM1504I S **Argument number** *argument-number* **in ENTRY reference** *ENTRY name* **has type POINTER, which is invalid for an OFFSET parameter without an AREA qualifier.**

Explanation: POINTER expressions can be converted to OFFSET only if the OFFSET is declared with an AREA qualifier.

IBM1505I S **Argument number** *argument-number* **in ENTRY reference** *ENTRY name* **has type POINTER, which is invalid for a POINTER parameter since the OFFSET argument is not an OFFSET variable declared with an AREA qualifier.**

Explanation: OFFSET variables can be converted to POINTER only if the OFFSET is declared with an AREA qualifier.

IBM1506I S **Argument number** *argument-number* **in ENTRY reference** *ENTRY name* **has a different ORDINAL type than the corresponding parameter.**

Explanation: ORDINALs cannot be passed to other ORDINALs having different ORDINAL types.

IBM1507I S **Arrays of label constants may not be passed as arguments.**

Explanation: The array can be assigned to an array of LABEL variables, and that array can be passed.

```
lx(1): ... ;

lx(2): ... ;

call x( lx );
```

IBM1508I S **Too few arguments have been specified for the ENTRY** *ENTRY name*.

Explanation: The number of arguments must match the number of parameters in the ENTRY declaration.

IBM1509I S **Argument to** *variable name* **pseudovariable must be ASSIGNABLE.**

Explanation: The target in an assignment through a pseudovariable must not have the NONASSIGNABLE attribute.

```
dcl a static nonasgn char(7) init('example');

unspec(a) = ''b;
```

IBM1510I S **First argument to** *variable name* **pseudovariable must be ASSIGNABLE.**

Explanation: The target in an assignment through a pseudovariable must not have the NONASSIGNABLE attribute.

```
dcl a static nonasgn char(7) init('example');

substr(a,1,2) = 'tr';
```

© Copyright IBM Corp. 1999, 2018

IBM1511I S **Argument number** *argument-number* **in ENTRY reference** *ENTRY name* **is an aggregate, but the parameter description specifies a scalar.**

Explanation: Scalars cannot be converted to aggregates.

```
dcl a entry( fixed bin ), b(10) fixed bin;

call a( b );
```

IBM1512I S **Argument number** *argument-number* **in ENTRY reference** *ENTRY name* **is a scalar, but the parameter description specifies an aggregate to which it cannot be passed.**

Explanation: Dummy aggregate arguments are not supported except when passing a non-AREA scalar to a non-CONTROLLED array of scalars, and the array must have no bounds specified as *. The scalar can be assigned to an aggregate, and that aggregate can be passed.

```
dcl a entry( 1, 2 fixed bin, 2 fixed bin );

call a( 0 );
```

IBM1513I S **Argument number** *argument-number* **in ENTRY reference** *ENTRY name* **is an aggregate that does not exactly match the corresponding parameter description.**

Explanation: Dummy aggregate arguments are not supported. If an entry description describes an aggregate parameter, then any argument passed must match that parameter's description.

IBM1514I S **Argument number** *argument-number* **in ENTRY reference** *ENTRY name* **is an aggregate with more members than its corresponding parameter description.**

Explanation: Dummy aggregate arguments are not supported. If an entry description describes an aggregate parameter, then any argument passed must match that parameter's description.

IBM1515I S **Argument number** *argument-number* **in ENTRY reference** *ENTRY name* **is an aggregate with fewer members than its corresponding parameter description.**

Explanation: Dummy aggregate arguments are not supported. If an entry description describes an aggregate parameter, then any argument passed must match that parameter's description.

IBM1516I S **The number of dimensions in the subelements of argument number** *argument-number* **in ENTRY reference** *ENTRY name* **and in its corresponding parameter description do not match.**

Explanation: Dummy aggregate arguments are not supported. If an entry description describes an aggregate parameter, then any argument passed must match that parameter's description.

IBM1517I S **The upper and lower bounds in the subelements of argument number** *argument-number* **in ENTRY reference** *ENTRY name* **and in its corresponding parameter description do not match.**

Explanation: Dummy aggregate arguments are not supported. If an entry description describes an aggregate parameter, then any argument passed must match that parameter's description.

IBM1518I S **The number of dimensions for argument number** *argument-number* **in ENTRY reference** *ENTRY name* **and in its corresponding parameter description do not match.**

Explanation: Array arguments and parameters must have the same number of dimensions.

```
dcl a entry( (*,*) fixed bin ), b (10) fixed bin;

call a( b );
```

IBM1519I S **The upper and lower bounds for argument number** *argument-number* **in ENTRY reference** *ENTRY name* **and in its corresponding parameter description do not match.**

Explanation: Array arguments and parameters must have the same lower and upper bounds.

```
dcl a entry( (0:10) fixed bin ), b (10) fixed bin;

call a( b );
```

IBM1520I S **Charset 48 is not supported.**

Explanation: Charset 48 is no longer supported. The source code must be converted to charset 60.

IBM1521I S **Not enough virtual memory is available to continue the compilation.**

Explanation: The compilation requires more virtual memory than is available. It may help to specify one or more of the following compiler options: NOTEST,

NOXREF, NOATTRIBUTES, and NOAGGREGATE.

IBM1522I S *variable* **cannot be SET unless an IN clause is specified.**

Explanation: If an offset variable is declared without an AREA reference, it cannot be set in an ALLOCATE or LOCATE statement unless an IN clause names an AREA reference.

IBM1523I S **Argument to** *BUILTIN name* **built-in must be an AREA reference.**

Explanation: The built-in function AVAILABLEAREA is defined only for AREAs.

IBM1524I S *BUILTIN name***(x) is undefined if ABS(x) > 1.**

Explanation: An expression contains the built-in function ASIN or ACOS applied to a restricted expression that evaluated to a number outside the domain of that function.

IBM1525I S **ATANH(x) is undefined if x is REAL and ABS(x) >= 1.**

Explanation: An expression contains the built-in function ATANH applied to a restricted expression that evaluated to a number outside the domain of that function.

IBM1526I S **Argument to** *BUILTIN name* **must have derived mode REAL.**

Explanation: An expression contains the named built-in function with an argument having mode COMPLEX.

IBM1527I S **First argument to** *BUILTIN name* **built-in must have locator type.**

Explanation: An expression contains the named built-in function with its first argument having neither type POINTER nor OFFSET.

IBM1528I S **First argument to** *BUILTIN name* **built-in must have derived mode REAL.**

Explanation: An expression contains the named built-in function with its first argument having mode COMPLEX. This message applies, for example, to the ATAN and ATAND built-in functions when two arguments are given.

IBM1530I S **Second argument to** *BUILTIN name* **built-in must have derived mode REAL.**

Explanation: An expression contains the named built-in function, with its second argument having mode COMPLEX. This message applies, for example, to the ATAN and ATAND built-in functions when two arguments are given.

IBM1531I S *BUILTIN name* **argument has invalid type.**

Explanation: An expression contains the reference BINARYVALUE(x) where x has a type other than POINTER, OFFSET or ORDINAL.

IBM1532I S **E35 sort exit routines must use a 32-bit linkage.**

Explanation: Any other linkage is invalid.

IBM1533I S *BUILTIN name* **argument must have computational type.**

Explanation: An expression contains the named built-in function with an argument that has neither string nor numeric type.

IBM1534I S *BUILTIN name* **result would be too long.**

Explanation: The result of the REPEAT or COPY built-in function must not be longer than the maximum allowed for the base string type.

IBM1535I S *BUILTIN name* **argument must have type REAL FLOAT.**

Explanation: An expression contains the named built-in function with an argument having type other than REAL FLOAT. This message applies, for instance, to the floating-point inquiry built-in functions such as HUGE and RADIX, and to the floating-point manipulation built-in functions such as EXPONENT and SUCC.

IBM1536I S *BUILTIN name* **argument must be a reference.**

Explanation: An expression contains the named built-in function with an argument that is not a reference.

IBM1537I S *BUILTIN name* **argument must be an array expression.**

Explanation: An expression contains the named built-in function with an argument that is not an array expression. This message applies, for example, to the built-in functions ALL, ANY, SUM and PROD.

IBM1538I S *BUILTIN name* **argument must be a FILE reference.**

Explanation: An expression contains the named built-in function with an argument that is not a FILE. This message applies, for example, to the I/O built-in functions such as LINENO and PAGENO.

IBM1539I S *** is invalid as a BUILTIN function argument.**

Explanation: A value must be specified as an argument to a BUILTIN function unless the argument is optional.

```
dcl a float;

a = sqrt(*);
```

IBM1540I S **Argument number** *argument number* **to** *BUILTIN name* **built-in must have derived mode REAL.**

Explanation: An expression contains the named built-in function with the specified argument having mode COMPLEX. This message applies to the MAX and MIN built-in functions.

IBM1541I S **Argument number** *argument number* **to** *BUILTIN name* **built-in must have computational type.**

Explanation: An expression contains the named built-in function with the specified argument having noncomputational type. This message applies to the MAX and MIN built-in functions.

IBM1542I S **First argument to** *BUILTIN name* **built-in must have computational type.**

Explanation: An expression contains the named built-in function with a first argument that has neither string nor numeric type.

IBM1543I S **Argument to** *BUILTIN name* **built-in must have type CHARACTER(1) NONVARYING.**

Explanation: This applies to the RANK built-in function.

IBM1545I S **First argument to** *BUILTIN name* **built-in must be an array.**

Explanation: An expression contains the named built-in function with a first argument that is not an array. This message applies, for instance, to the DIMENSION, HBOUND, and LBOUND built-in functions.

IBM1546I S **Second argument to** *BUILTIN name* **built-in must have type CHARACTER(1) NONVARYING.**

Explanation: This applies to the PLIFILL built-in subroutine.

IBM1547I S **Second argument to** *BUILTIN name* **built-in must have computational type.**

Explanation: An expression contains the named built-in function with a second argument that has neither string nor numeric type.

IBM1548I S *BUILTIN function* **may not be used inside a BEGIN block.**

Explanation: The PLISTSIZE built-in functions may be used only in procedures.

IBM1549I S *BUILTIN function* **may be used only in procedures with LINKAGE(SYSTEM).**

Explanation: The PLISTSIZE built-in function may not be used in procedures with any of the linkages OPTLINK, PASCAL, etc.

IBM1550I S **Argument to the** *BUILTIN name* **pseudovariable must be an EVENT variable.**

Explanation: This message applies to the COMPLETION and STATUS pseudovariables.

IBM1551I S **Argument to the** *BUILTIN name* **pseudovariable must be a TASK variable.**

Explanation: This message applies to the PRIORITY pseudovariable.

IBM1552I S **Third argument to** *BUILTIN name* **built-in must have computational type.**

Explanation: An expression contains the named built-in function with a third argument that has neither string nor numeric type. This message applies, for example, to the SUBSTR and CENTER built-in functions.

IBM1554I S **Argument to** *BUILTIN name* **built-in must be either a NONVARYING BIT array reference or else an array expression with known length.**

Explanation: The ALL and ANY built-in functions are restricted to two types of array expressions: an array expression that is a NONVARYING BIT array reference or an array expression that has known length. The first

five examples below meet these restrictions, but the remaining examples do not.

```
dcl a(10) bit(16) varying;
dcl b(10) bit(16);

if all( b ) then ...
if any( a ^= ''b ) then ...
if all( a = b & a ) then ...
if any( ''b ^= b ) then ...
if all( a = ''b | b = ''b ) then ...
if any( a ) then ...
if all( substr(b,1,n) ) then ...
```

IBM1555I S **Second argument to** *BUILTIN name* **built-in must have computational type.**

Explanation: An expression contains the named built-in function with a second argument that has neither string nor numeric type.

IBM1556I S **Argument number** *argument number* **to** *BUILTIN name* **built-in would force STRINGRANGE.**

Explanation: If a third argument is given for one of the built-in functions INDEX, SEARCH, VERIFYR, or SCRUBOUT, it must be positive. If a third argument is given for one of the built-in functions SEARCHR and VERIFYR, it must be nonnegative. If a fourth argument is given for the built-in function REPLACE, it must be positive.

IBM1557I S **Second argument to** *BUILTIN name* **built-in must be positive.**

Explanation: The second argument for the built-in functions CENTER, LEFT and RIGHT must not be zero or negative.

IBM1558I S **Argument to VALID built-in must have the attributes FIXED DECIMAL or PICTURE.**

Explanation: The argument to the VALID built-in function must have exactly the indicated attributes. It is not sufficient that it can be converted to these attributes.

IBM1559I S **SQRT(x) is undefined if x is REAL and x < 0.**

Explanation: An expression contains the BUILTIN function SQRT applied to a restricted expression that evaluated to a number outside the domain of that function.

IBM1560I S *BUILTIN function***(x) is undefined if x is REAL and x <= 0.**

Explanation: An expression contains the named built-in function applied to a restricted expression that evaluated to a number outside the domain of that function. This message applies, for instance, to the LOG, LOG2, and LOG10 built-in functions.

IBM1561I S **RULES(ANS) does not allow ROUND to be applied to FIXED BIN.**

Explanation: RULES(ANS) dose not permit non-zero scale factors with FIXED BIN, and hence it does not allow ROUND to be applied to FIXED BIN (or BIT) arguments.

IBM1562I S **Argument to** *BUILTIN name* **built-in has invalid type.**

Explanation: The argument to the HANDLE built-in must be a structure type, and conversely the argument to the TYPE built-in must be a handle.

IBM1563I S **Second argument to** *BUILTIN name* **built-in must be nonnegative.**

Explanation: The second argument for the built-in functions CHARACTER, BIT, and GRAPHIC must be zero or greater.

IBM1564I S **Too few arguments have been specified for the** *BUILTIN name* **built-in.**

Explanation: Supply the minimum number of arguments required.

IBM1566I S *BUILTIN name***(x) is undefined for x outside the supported domain.**

Explanation: An expression contains the named built-in function applied to a restricted expression that evaluated to a number outside the supported domain of that function.

IBM1568I S *BUILTIN function***(x,y) is undefined if x=0 and y=0.**

Explanation: An expression contains the built-in function ATAN or ATAND applied to a restricted expression that evaluated to a number outside the domain of that function.

IBM1569I S *BUILTIN name* **argument must be a CONNECTED reference.**

Explanation: The argument to the named built-in function must be a reference (for example, not an expression or a literal), and that reference must be CONNECTED.

IBM1570I S *BUILTIN name* **argument must be a reference to a level 1 CONTROLLED variable.**

Explanation: The ALLOCATION built-in function cannot be used with structure members or with non-CONTROLLED variables.

IBM1571I S *BUILTIN name* **argument must be a reference to a level 1 BYADDR parameter.**

Explanation: The OMITTED built-in function cannot be used with BYVALUE parameters, structure members, or non-parameters.

IBM1573I S **The use of * as an argument is permitted only for parameters declared with the OPTIONAL attribute.**

Explanation: Add the OPTIONAL attribute to the entry declaration or replace the * by an actual argument.

IBM1575I S **Argument number** *argument number* **to** *BUILTIN name* **built-in must have type POINTER or OFFSET.**

Explanation: The indicated argument to built-in functions such as PLIMOVE and COMPARE must be a locator.

IBM1576I S **Argument number** *argument number* **to** *BUILTIN name* **built-in must have type CHARACTER(1) NONVARYING.**

Explanation: This applies to HEXIMAGE, CENTER, LEFT, RIGHT, MEMSQUEEZE, etc.

IBM1577I S **First argument to** *BUILTIN name* **built-in must have type POINTER.**

Explanation: This applies to the OFFSET built-in function.

IBM1578I S **First argument to** *BUILTIN name* **built-in must have type OFFSET.**

Explanation: This applies to the POINTER built-in function.

IBM1579I S **Second argument to** *BUILTIN name* **built-in must have type AREA.**

Explanation: This applies to the OFFSET and POINTER built-in functions.

IBM1580I S **First argument to** *BUILTIN name* **built-in is an OFFSET value.**

Explanation: If the first argument to built-in functions such as PLIMOVE and COMPARE has the attribute OFFSET, it must be an OFFSET reference not an OFFSET value.

IBM1581I S **First argument to** *BUILTIN name* **built-in is an OFFSET variable declared without an AREA qualifier.**

Explanation: If the first argument to built-in functions such as PLIMOVE and COMPARE is an OFFSET variable, that OFFSET variable must be declared with an AREA qualifier so that the offset can be converted to an address.

IBM1582I S **Argument number** *argument number* **to** *BUILTIN name* **built-in is an OFFSET value.**

Explanation: If the indicated argument to built-in functions such as PLIMOVE and COMPARE has the attribute OFFSET, it must be an OFFSET reference not an OFFSET value.

IBM1583I S **Argument number** *argument number* **to** *BUILTIN name* **built-in is an OFFSET variable declared without an AREA qualifier.**

Explanation: If the indicated argument to built-in functions such as PLIMOVE and COMPARE is an OFFSET variable, that OFFSET variable must be declared with an AREA qualifier so that the offset can be converted to an address.

IBM1584I S **Second argument to** *BUILTIN name* **built-in must have type OFFSET.**

Explanation: This applies to the OFFSETDIFF built-in function.

IBM1585I S **Second argument to** *BUILTIN name* **built-in must have type POINTER.**

Explanation: This applies to the POINTERDIFF built-in function.

IBM1586I S **Argument to STRING built-in function/pseudovariable must be CONNECTED.**

Explanation: The STRING built-in function and pseudovariable cannot be applied to discontiguous array cross-sections or to array parameters not declared with the CONNECTED attribute.

IBM1587I S **Argument number** *argument number* **to** *BUILTIN name* **built-in must have the ENTRY attribute.**

Explanation: Any other argument type is invalid. This message applies to the PLISRTx built-in functions.

IBM1588I S **First argument to** *BUILTIN name* **built-in must have type GRAPHIC.**

Explanation: This applies to the CHARGRAPHIC built-in function. For instance, in the following example, g should be declared as graphic, not as char.

```
dcl c char(10);
dcl g char(5);

c = charg( g );
```

IBM1589I S *BUILTIN name* **argument must not have any subscripts.**

Explanation: The LOCATION and BITLOCATION built-in functions cannot be applied to subscripted references.

IBM1590I S **Argument to STRING built-in function/pseudovariable must not be a UNION and must not contain a UNION.**

Explanation: The STRING built-in function and pseudovariable cannot be applied to UNIONs or to structures containing UNIONs.

IBM1591I S **All members of an argument to the STRING built-in function/ pseudovariable must have the UNALIGNED attribute.**

Explanation: The STRING built-in function and pseudovariable cannot be applied to structures or arrays containing elements with the ALIGNED attribute.

IBM1592I S **All members of an argument to the STRING built-in function/ pseudovariable must have the NONVARYING attribute.**

Explanation: The STRING built-in function and pseudovariable cannot be applied to structures or arrays containing VARYING strings.

IBM1593I S **All members of an argument to the STRING built-in function/ pseudovariable must have string type.**

Explanation: The STRING built-in function and pseudovariable cannot be applied to structures or arrays containing noncomputational types or arithmetic types other than pictures.

IBM1594I S **All members of an argument to the STRING built-in function/ pseudovariable must have the same string type.**

Explanation: The STRING built-in function and pseudovariable cannot be applied to structures or arrays containing different string types, for example, BIT and CHARACTER strings.

IBM1595I S **First argument to** *BUILTIN name* **built-in must have type REAL FLOAT.**

Explanation: This applies to the floating-point inquiry and manipulation built-in functions such as HUGE and EXPONENT.

IBM1596I S **Second argument to** *BUILTIN name* **built-in must have type CHARACTER.**

Explanation: This applies to the EDIT built-in function.

IBM1597I S *BUILTIN name* **argument must have type TASK.**

Explanation: This applies to the PRIORITY built-in function.

IBM1598I S *BUILTIN name* **argument must have type EVENT.**

Explanation: This applies to the COMPLETION and STATUS built-in functions.

IBM1599I S **The BUILTIN function** *variable name* **may not be used as a pseudovariable.**

Explanation: The named built-in function is not a pseudovariable and may not be used as one.

IBM1600I S **Source to** *BUILTIN name* **pseudovariable must be scalar.**

Explanation: It is invalid to assign an array, structure, or union to one of the built-in functions ONCHAR, ONSOURCE, or ONGSOURCE.

IBM1601I S **The identifier** *identifier* **is not the name of a built-in function. Any use of it is unsupported.**

Explanation: The BUILTIN attribute can be applied only to identifiers that are the names of built-in functions or subroutines.

IBM1602I S **Fourth argument to** *BUILTIN name* **built-in must have the attributes REAL FIXED BIN(31,0).**

Explanation: This applies to the PLISRTx built-in functions. For instance, in the following example, rc should be declared as fixed bin(31), not fixed bin(15).

```
dcl rc fixed bin(15);

call plisrta( 'SORT FIELDS=(1,80,CH,A) ',
              'RECORD  TYPE=F,LENGTH=(80) ',
              256000,
              rc );
```

IBM1603I S *BUILTIN name* **argument must not have the CONSTANT attribute.**

Explanation: This applies to the ADDR and similar built-in functions. It is invalid, for instance, to apply the ADDR built-in function to a label constant.

IBM1604I S *BUILTIN function* **argument must be nonnegative.**

Explanation: The argument for the built-in functions LOW and HIGH must be zero or greater.

IBM1605I S **Argument to ENTRYADDR built-in must be an ENTRY variable or an EXTERNAL ENTRY constant.**

Explanation: The ENTRYADDR built-in function cannot be applied to non-ENTRYs or to INTERNAL ENTRY constants.

IBM1606I S **Argument to** *variable name* **pseudovariable must be a reference.**

Explanation: Pseudovariables cannot be applied to expressions.

```
unspec( 12 ) = '00'b4;
```

IBM1607I S **First argument to** *variable name* **pseudovariable must be a reference.**

Explanation: The SUBSTR pseudovariable cannot be applied to expressions.

```
substr( 'nope', 1, 1 ) = 'd';
```

IBM1608I S **Argument to** *variable name* **pseudovariable must be a scalar.**

Explanation: The compiler does not support the named pseudovariable applied to arrays, structures, or unions.

IBM1609I S **First argument to** *variable name* **pseudovariable must be a scalar.**

Explanation: The compiler does not support the named pseudovariable applied to arrays, structures, or unions.

IBM1610I S **Argument to** *variable name* **pseudovariable must be COMPLEX.**

Explanation: The REAL and IMAG pseudovariable can be applied only to COMPLEX arithmetic variables.

IBM1611I S **First argument to SUBSTR pseudovariable must have string type.**

Explanation: The SUBSTR pseudovariable cannot be applied to numeric variables or to noncomputational values.

IBM1612I S **Argument to the ENTRYADDR pseudovariable must be an ENTRY variable.**

Explanation: The ENTRYADDR pseudovariable can be applied only to ENTRY variables.

IBM1613I S **Argument to** *BUILTIN name* **built-in has attributes that conflict with** *file attribute***.**

Explanation: The indicated built-in function cannot be applied to file constants with attributes that conflict with the indicated attribute.

IBM1614I S **Argument to** *BUILTIN name* **built-in has attributes that conflict with STREAM.**

Explanation: The indicated built-in function cannot be applied to non-STREAM files.

IBM1615I S **Argument to** *BUILTIN name* **built-in has attributes that conflict with PRINT.**

Explanation: The indicated built-in function cannot be applied to non-PRINT files.

IBM1616I S **Attributes and ENVIRONMENT options for file** *file name* **conflict.**

Explanation: Specified file attributes and ENVIRONMENT options on a declaration statement are in conflict. The following DECLARE statement is an example of this type of conflict:

```
dcl file f1 direct env(consecutive);
```

IBM1617I S DIRECT attribute for file *file name* needs ENVIRONMENT option specification of INDEXED, REGIONAL, RELATIVE, or VSAM.

Explanation: Use of the DIRECT file attribute needs an ENVIRONMENT option specification of INDEXED, REGIONAL, RELATIVE, or VSAM.

```
dcl file f1 direct env(relative);
```

IBM1618I S Syntax of the %INCLUDE statement is incorrect.

Explanation: %INCLUDE must be followed by a name and either a semicolon or else a second name in parenthesis and then a semicolon.

IBM1619I S File specification after %INCLUDE is too long.

Explanation: The maximum length of the file specification is 8 characters.

IBM1620I S File specification missing after %INCLUDE.

Explanation: %INCLUDE must be followed by a file name, not just a semicolon.

IBM1621I S NODESCRIPTOR attribute is invalid if any parameters have bit alignment.

Explanation: If a parameter is an unaligned bit string or an array or structure consisting entirely of unaligned bit strings, then OPTIONS(NODESCRIPTOR) must not be specified or implied.

IBM1622I S The number of elements and dimension specifications in an aggregate must not exceed 131071.

Explanation: Aggregates with more than 131071 elements and dimension specifications would require descriptors that would require too much storage.

IBM1623I S The dot-qualified reference *reference name* is unknown.

Explanation: The named reference is not a member of any structure or union declared in the block in which it is referenced or declared in any block containing that block.

IBM1625I S Extent must be a scalar.

Explanation: An expression specifying an array bound, a string length or an AREA size must not be a reference to an array, a structure, or a union.

IBM1626I S Extent must have computational type.

Explanation: An expression specifying an array bound, a string length, or an AREA size must have numeric or string type.

IBM1627I S Subscript expressions must be scalars.

Explanation: An expression used as a subscript must not be an array, structure, or union reference.

IBM1628I S Index number *index number* into the array *variable name* must have computational type.

Explanation: Only expressions having numeric or string type may be used as subscripts.

IBM1629I S Extents for STATIC variable are not constant.

Explanation: Array bounds, string lengths, and AREA sizes in STATIC variables must evaluate at compile-time to constants.

IBM1630I S Number of dimensions in arrays do not match.

Explanation: In the assignment of one array to another, the two arrays must have the same number of dimensions.

IBM1631I S Upper and lower bounds in arrays do not match.

Explanation: In the assignment of one array to another, the two arrays must have the same lower and upper bound in each dimension.

IBM1632I S Index number *index number* into the variable *variable name* is less than the lower bound for that dimension.

Explanation: Executing such a program would most likely cause a protection exception.

```
dcl a(5:10)  fixed bin(31);

a(1) = 0;
```

IBM1633I S Index number *index number* into the variable *variable name* is greater than the upper bound for that dimension.

Explanation: Executing such a program would most likely cause a protection exception.

```
dcl a(5:10)  fixed bin(31);

a(20) = 0;
```

IBM1634I S Number of dimensions in subelements of structures do not match.

Explanation: In structure assignments and structure expressions, all subelements that are arrays must have the same number of dimensions.

```
dcl
  1 a,
    2 b(8)     fixed bin,
    2 c        char(10);

dcl
  1 x,
    2 y(8,9)   fixed bin,
    2 z        char(10);

a = x;
```

IBM1635I S Upper and lower bounds in subelements of structures do not match.

Explanation: In structure assignments and structure expressions, all subelements that are arrays must have the same bounds.

```
dcl
  1 a,
    2 b(8)     fixed bin,
    2 c        char(10);

dcl
  1 x,
    2 y(9)     fixed bin,
    2 z        char(10);

a = x;
```

IBM1636I S Substructuring in subelements of structures do not match.

Explanation: In structure assignments and structure expressions, if any element of one structure is itself a structure, then the corresponding element in all the other structures must also be a similar structure.

IBM1637I S Number of subelements in structures do not match.

Explanation: In structure assignments and structure expressions, all structures must have the same number of elements.

IBM1638I S Structures and unions are not permitted in GENERIC descriptions.

Explanation: Only scalars and arrays of scalars are permitted in GENERIC descriptions.

IBM1639I S The aggregate *aggregate-name* contains only noncomputational values. The aggregate will be ignored.

Explanation: Aggregates containing no strings or arithmetic variables cannot be used in PUT or GET statements.

IBM1640I S The aggregate *aggregate-name* contains one or more unions and cannot be used in stream I/O.

Explanation: Aggregates containing one or more UNION statements cannot be used in PUT or GET statements.

IBM1641I S References to slices of the array of structures *structure-name* are not permitted.

Explanation: An array of structures must be referenced in its entirety or element by element.

```
dcl
  1 a(8,9),
    2 b        fixed bin,
    2 c        char(10);

a(2,*) = 0;
```

IBM1642I S References to slices of the array of unions *union-name* are not permitted.

Explanation: An array of unions must be referenced in its entirety or element by element.

```
dcl
  1 a(8,9) union,
    2 b        fixed bin,
    2 c        char(10);

a(2,*) = 0;
```

IBM1643I S Each dimension of an array must contain no more than 2147483647 elements.

Explanation: It must be possible to compute the value of the DIMENSION built-in function for an array. In DECLARE x(x:y), (y-x+1) must be less than 214748648.

IBM1644I S Aggregate contains more than 15 logical levels.

Explanation: The maximum physical level allowed is 255, but the maximum logical level is 15.

IBM1645I S Data aggregate exceeds the maximum length.

Explanation: Aggregates containing unaligned bits must be less than 2**28 bytes in size while all other aggregates must be less than 2**31.

IBM1646I S SIZE would be raised in assigning TO value to control variable.

Explanation: If the TO value is bigger than the maximum value that a FIXED or PICTURE variable can hold, then a loop dominated by that variable would cause SIZE to be raised. For example, in the first code fragment below, x can not be assigned a value bigger than 99. In the second code fragment below, y can not be assigned a value bigger than 32767.

```
dcl x pic'99';

do x = 1 to 100;
  put skip list( x );
end;

dcl y fixed bin(15);

do y = 1 to 32768;
  put skip list( y );
end;
```

IBM1647I S Too few subscripts specified for the variable *variable name***.**

Explanation: The number of subscripts given for a variable must match that variable's number of dimensions

IBM1648I S Too many subscripts specified for the variable *variable name***.**

Explanation: The number of subscripts given for a variable must match that variable's number of dimensions

IBM1649I S The number of inherited dimensions plus the number of member dimensions exceeds 15.

Explanation: Arrays with more than 15 dimensions are not supported.

```
dcl
  1 dim7(2,3,4,5,6,7,8),
    2 dim7more(2,3,4,5,6,7,8)
      3 dim2many(2,3)  fixed bin,
      3 *              fixed bin,
    2 *  char(10);
```

IBM1650I S The LIKE reference is neither a structure nor a union.

Explanation: The LIKE reference cannot be a scalar or an array of scalars.

```
dcl
  a fixed bin,
  1 b like a;
```

IBM1651I S The LIKE reference is ambiguous.

Explanation: The LIKE reference needs enough qualification to be unique.

```
dcl
  1 x like b,
  1 a,
    2 b,
      3 c,
      3 d,
    2 e,
      3 f,
      3 g,
  1 h,
    2 b,
      3 j,
      3 k;
```

IBM1652I S Neither the LIKE reference nor any of its substructures can be declared with the LIKE attribute.

Explanation: LIKE from LIKE is not supported.

```
dcl
  1 a,
    2 b1 like c,
    2 b2 like c,
  1 c,
    2 d fixed bin,
    2 e fixed bin;
dcl
  1 x like a;
```

IBM1653I S The LIKE reference must not be a member of a structure or union declared with the LIKE attribute.

Explanation: LIKE from LIKE is not supported.

```
dcl
  1 a,
    2 b1 like c,
```

```
    2 b2 like c,
  1 c,
    2 d fixed bin,
    2 e fixed bin;
dcl
  1 x like a.b1;
```

IBM1654I S The LIKE reference is unknown.

Explanation: The LIKE reference must be known in the block containing the LIKE attribute specification.

IBM1655I S Only CONTROLLED variables can be passed to CONTROLLED parameters.

Explanation: If a parameter is declared as controlled, non-controlled variables and expressions with operators cannot be passed to it.

```
dcl c char(20);

call a(c);

a: proc( b );
  dcl b controlled char(*);
```

IBM1656I S A CONTROLLED variable passed to a CONTROLLED parameter must have the same attributes as that parameter.

Explanation: Differences in any arithmetic attributes are not permitted. The following example will emit this message.

```
dcl x fixed bin(15) controlled;

call a(x);

a: proc( b );
  dcl b controlled fixed bin(31);
```

IBM1657I S A subscript has been specified for the non-array variable *variable name***.**

Explanation: Subscripts are permitted only in array element references.

IBM1658I S Argument number *argument-number* **in ENTRY reference** *ENTRY name* **is an array expression requiring a temporary array with strings of unknown length.**

Explanation: Temporary arrays of strings are supported only if the string length is known.

```
dcl a entry, (b(10),c(10)) char(20) var;

call a( b || c );
```

IBM1659I S After LIKE expansion, aggregate would contain more than 15 logical levels.

Explanation: The total number of logical levels after LIKE expansion must not exceed 15.

IBM1660I S The size (*record-size* **) of the record conflicts with the RECSIZE (** *recsize* **) specified in the ENVIRONMENT attribute.**

Explanation: Execution of the statement would raise the RECORD condition.

```
dcl datei         file record output
                  env( fb recsize (80) total ) ;

dcl satzaus       char (100);

write file(datei) from(satzaus);
```

IBM1661I S Aggregates cannot be assigned to scalars.

Explanation: Only scalars can be assigned to scalars.

IBM1662I S Unsupported use of union or structure containing a union.

Explanation: Unions and structures containing unions may not be used in expressions except when used as an argument to a built-in function such as ADDR or UNSPEC.

IBM1663I S Unsupported or invalid use of structure expression.

Explanation: Structure expressions may not, for instance, be assigned to arrays of scalars.

IBM1664I S Array expressions cannot be assigned to non-arrays.

Explanation: Array expressions may not, for instance, be assigned to structures or scalars.

IBM1665I S E15 sort exit routines must have the RETURNS attribute.

Explanation: An E15 sort exit have the RETURNS attribute since it will be invoked as a function by the sort library routine.

IBM1666I S E15 sort exit routines must return a CHARACTER string.

Explanation: An E15 sort exit may return a NONVARYING, VARYING or VARYINGZ CHARACTER string, but it must be a character string.

IBM1667I S Target in assignment is NONASSIGNABLE.

Explanation: The target in an assignment statement must not have the NONASSIGNABLE attribute.

IBM1668I S Target in assignment is a function reference.

Explanation: The target of an assignment statement must be an array, structure, union or scalar reference. Function references are not permitted as target of assignments.

IBM1669I S Unsupported assignment to a target containing a UNION.

Explanation: Assignments to UNIONs or structures containing UNIONs are restricted. Compound assignment operators are not supported, the source must be a similar structure that contains matching UNIONs, both the source and target must have extents known at compile time, and all UNIONs involved must occupy a whole number of bytes.

IBM1670I S A PROCEDURE containing ENTRY statements with differing RETURNS attributes must return values BYADDR.

Explanation: In a PROCEDURE containing ENTRY statements, if the PROCEDURE and ENTRY statements do not all have the same RETURNS attributes, then all values must be returned BYADDR. You can compile with DFT(RETURNS(BYADDR)) to force this, or you can add the BYADDR attribute to each set of RETURNS attribute. For example, you must either compile the following program with DFT(RETURNS(BYADDR)) or change the "fixed bin" to "fixed bin byaddr".

```
a: proc;
  return;
b: entry returns( fixed bin );
  return( 1729 );
end;
```

IBM1671I S The source in a structure assignment must be a scalar expression or a matching structure.

Explanation: The source in a structure assignment cannot be an array of scalars or a structure that does not match the target.

IBM1672I S In multiple BY NAME assignments, if one target is an array of structures, then all must be.

Explanation: A BY NAME assignment may have not have a mixture of array and non-array targets.

```
dcl 1 a, 2 a1 fixed bin, 2 a2 fixed bin;
dcl 1 b(3), 2 a1 fixed bin, 2 a2 fixed bin;
dcl 1 c, 2 a1 fixed bin, 2 a2 fixed bin;

a,b = c, by name;
```

IBM1673I S The target in a compound concatenate and assign must be a VARYING or VARYINGZ string.

Explanation: Only the simple assignment operator can be used to assign to a NONVARYING string.

IBM1674I S Target in assignment contains UNIONs.

Explanation: The target in an assignment must not contain any UNIONs.

IBM1675I S FROMALIEN option cannot be used with MAIN.

Explanation: These two options are mutually exclusive.

IBM1676I S Source in assignment to LIMITED ENTRY must be either a non-nested ENTRY constant or another LIMITED ENTRY.

Explanation: ENTRY constants representing nested procedures and ENTRY variables not declared with the LIMITED attribute cannot be assigned to variables with the attributes LIMITED ENTRY.

IBM1677I S Assignment of ENTRY to *target type* is invalid. If the ENTRY should be invoked, an argument list must be provided.

Explanation: An ENTRY constant or variable without an argument list will not be invoked and hence can be assigned only to an ENTRY variable.

IBM1678I S Assignment of *source type* to *target type* is invalid.

Explanation: The target attributes conflict with the source attributes.

IBM1679I S Assignment of POINTER to OFFSET is invalid unless the OFFSET is declared with an AREA qualifier.

Explanation: POINTER expressions can be converted to OFFSET only if the OFFSET is declared with an AREA qualifier.

IBM1680I S Assignment of OFFSET to POINTER is invalid unless the OFFSET is declared with an AREA qualifier.

Explanation: OFFSET variables can be converted to POINTER only if the OFFSET is declared with an AREA qualifier.

IBM1681I S The number of preprocessor invocations specified exceeds the maximum number (25) allowed.

Explanation: A maximum of 25 preprocessor invocations can be specified in the PP option or in combination with the MACRO option.

IBM1682I S The target in a BY NAME assignment must be a structure.

Explanation: The target in a BY NAME assignment cannot be an array or a scalar.

IBM1683I S Set of matching names in the expansion of BY NAME assignment must contain either all structures or no structures.

Explanation: For instance, in the assignment, x = y, by name, if both x and y immediately contain a member z, then either both x.z and y.z are structures or neither x.z and y.z is a structure.

IBM1684I S Number of dimensions in the BY NAME corresponding elements *variable name* **and** *variable name* **do not match.**

Explanation: In a BY NAME assignment, arrays with matching names must have the same number of dimensions.

```
dcl
  1 a,
    2 b(4,5)  bin(31,0),
    2 c       bin(31,0);
dcl
  1 x,
    2 b(4)    bin(31,0),
    2 c       bin(31,0);

a = x, by name;
```

IBM1685I S Upper and lower bounds in BY NAME corresponding elements *variable name* **and** *variable name* **do not match.**

Explanation: In a BY NAME assignment, arrays with matching names must have the same lower and upper bounds.

```
dcl
  1 a,
    2 b(1:5)  bin(31,0),
    2 c       bin(31,0);
dcl
  1 x,
    2 b(0:4)  bin(31,0),
    2 c       bin(31,0);

a = x, by name;
```

IBM1686I S BY NAME assignment contains UNIONs.

Explanation: The target structure in a BY NAME assignment must not contain any UNIONs even if no names in those UNIONs match names in the source. The source expression also must contain any unions or structures containing unions.

IBM1687I S *reserved name* **cannot be declared with OPTIONS other than ASM.**

Explanation: If the DLI compiler option is specified, PLITDLI cannot be declared with any OPTIONS other than OPTIONS(ASM).

IBM1688I S *reserved name* **cannot be declared with an entry description list.**

Explanation: If the DLI compiler option is specified, PLITDLI cannot be declared with an entry description list.

IBM1689I S *reserved name* **cannot be declared as a function.**

Explanation: If the DLI compiler option is specified, PLITDLI cannot be declared as a function.

IBM1690I S OPTIONS(*language-name***) is not supported for functions.**

Explanation: Functions, i.e. entrys declared with the RETURNS attribute, cannot be declared with OPTIONS(ASM) or OPTIONS(COBOL).

IBM1691I S Extents in ENTRY descriptors must be asterisks or restricted expressions with computational type.

Explanation: In ENTRY descriptors, each array bound, string length and AREA size must be specified either with an asterisk or with a restricted expression that has computational type.

IBM1692I S An ENTRY invoked as a function must have the RETURNS attribute.

Explanation: There is no default RETURNS attribute.

```
dcl e entry;

a = e();
```

IBM1693I S ***call-option*** **option repeated in CALL statement.**

Explanation: The TASK, EVENT and PRIORITY options may be specified only once in any CALL statement.

IBM1694I S Reference in CALL statement must not be a built-in function.

Explanation: CALL x is invalid unless x is a built-in subroutine, an ENTRY constant, or an ENTRY variable. Built-in functions are not built-in references. For example, "Call SQRT(x)" is invalid.

IBM1695I S Reference in CALL statement must either be a built-in subroutine or have type ENTRY.

Explanation: CALL x is invalid unless x is a built-in subroutine, an ENTRY constant, or an ENTRY variable.

IBM1696I S RETURN statement without an expression is invalid inside a subprocedure that specified the RETURNS attribute.

Explanation: All RETURN statements inside functions must specify a value to be returned.

```
a: proc returns( fixed bin );

  return;
```

IBM1697I S RETURN statement is invalid inside a PROCEDURE that did not specify the RETURNS attribute.

Explanation: A statement of the form RETURN(x) is valid inside only PROCEDUREs that are defined with a RETURNS attribute.

IBM1698I S RETURN statement with an expression is invalid inside a BEGIN in a PROCEDURE that does not have the RETURNS(BYADDR) attribute.

Explanation: A statement of the form RETURN(x) is valid inside a BEGIN block only if the PROCEDURE enclosing that BEGIN block has the RETURNS(BYADDR) attribute explicitly or by default.

IBM1699I S Argument number ***argument-number*** **in ENTRY reference** ***ENTRY name*** **is an aggregate. This conflicts with the BYVALUE option.**

Explanation: Arrays, structures, and unions cannot be passed BYVALUE.

IBM1700I S AREAs must be passed BYADDR.

Explanation: Even AREA variables with constant size must be passed BYADDR.

IBM1701I S Argument number ***argument-number*** **in ENTRY reference** ***ENTRY name*** **is a string with unknown size. This conflicts with the BYVALUE option.**

Explanation: Only strings with constant size can be passed BYVALUE.

IBM1702I S The ***attribute keyword*** **attribute is invalid as a RETURNS subattribute.**

Explanation: Structures and union may not be returned. The following code example is invalid:

```
dcl a entry returns( 1 union, 2 ptr, 2 ptr );
```

IBM1703I S Reference in CALL statement must not be an aggregate reference.

Explanation: CALL references must be scalars.

```
dcl ea(10) entry;

call ea;
```

IBM1704I S Too many argument lists have been specified for the variable ***variable name*****.**

Explanation: A function can have only one argument list unless it returns an ENTRY, in which case it can have only two argument lists unless the returned ENTRY returns an ENTRY, and so on.

IBM1705I S RETURN expression with attribute ***source type*** **is invalid for RETURNS options specifying the attribute** ***target type*****.**

Explanation: The RETURN expression must have a type that can be converted to the type indicated in the RETURNS option.

```
a: proc returns( pointer )

  return( 0 );
end;
```

IBM1706I S RETURN expression with attribute *source type* is invalid for RETURNS options specifying the attribute *target type*. If the ENTRY should be invoked, an argument list must be provided.

Explanation: The RETURN expression must have a type that can be converted to the type indicated in the RETURNS option.

```
a: proc returns( pointer )

  dcl f entry returns( pointer );
  return( f );
end;
```

IBM1707I S RETURN expression with attribute *source type* is invalid for RETURNS options specifying the attribute LIMITED ENTRY.

Explanation: Only an EXTERNAL ENTRY CONSTANT, an ENTRY CONSTANT representing a non-nested PROCEDURE, or an ENTRY VARIABLE with the LIMITED attribute can be specified as the RETURNS expression in a function that returns a LIMITED ENTRY.

IBM1708I S RETURN expression with attribute POINTER is invalid for RETURNS options specifying the attribute OFFSET since the OFFSET attribute is not declared with an AREA qualifier.

Explanation: POINTER expressions can be converted to OFFSET only if the offset is declared with an AREA qualifier.

IBM1709I S RETURN expression with attribute OFFSET is invalid for RETURNS options specifying the attribute POINTER since the OFFSET expression is not an OFFSET variable declared with an AREA qualifier.

Explanation: OFFSET variables can be converted to POINTER only if the OFFSET is declared with an AREA qualifier.

IBM1710I S ORDINAL type in RETURN expression and RETURNS option must match.

Explanation: In a function that returns an ordinal, the ORDINAL type in any RETURN expression must be the same as returned by the function.

```
a: proc returns( ordinal color );

  dcl i ordinal intensity;
  return( i );
end;
```

IBM1711I S Expression in RETURN statement must be scalar.

Explanation: The expression in a RETURN statement must not be an array, a structure, or a union.

IBM1712I S External name specification must be a non-null string.

Explanation: EXTERNAL('') is invalid.

IBM1713I S Function *function name* contains no RETURN statement.

Explanation: Functions must contain at least one RETURN statement.

IBM1714I S Extents in RETURNS descriptors must be constants.

Explanation: In RETURNS descriptors, each array bound, string length, and AREA size must be specified with a restricted expression that has computational type. Unlike ENTRY descriptors, asterisks are not permitted.

IBM1715I S Exit from an ON-unit via RETURN is invalid.

Explanation: RETURN statements are not permitted in an ON-unit or any of its contained BEGIN blocks unless the contained block is also contained in a procedure defined in the ON-unit.

IBM1716I S FORMAT expression must be a scalar value.

Explanation: Expressions in FORMAT lists, including SKIP clauses, must represent scalar values.

IBM1717I S FORMAT expression must have computational type.

Explanation: Expressions in FORMAT lists, including SKIP clauses, must have computational type so that the expression can be converted to FIXED BIN(31).

IBM1718I S *source type* **is invalid as a Boolean expression.**

Explanation: The expression in an IF, WHILE, UNTIL, SELECT, or WHEN clause must have computational type so that it can be converted to BIT(1).

IBM1719I S **ENTRY is invalid as a Boolean expression. If an ENTRY should be invoked, an argument list must be provided.**

Explanation: The expression in an IF, WHILE, UNTIL, SELECT, or WHEN clause must have computational type so that it can be converted to BIT(1). An ENTRY cannot be used as a Boolean expression. If the ENTRY is a function which should be invoked, an argument list, even if it consists only of a left and right parenthesis, must be provided.

IBM1720I S **Expression for calculating size of variable with adjustable extents is too complicated. Variable may be defined in terms of itself.**

Explanation: An expression used in calculating the size of a variable must not depend on any values that the variable may have because those values do not exist until storage can be allocated for the variable.

IBM1721I S **Expression contains too many nested subexpressions.**

Explanation: The compiler's space for evaluating expressions has been exhausted. Rewrite the expression in terms of simpler expressions.

IBM1722I S **The number of error messages allowed by the MAXMSG option has been exceeded.**

Explanation: Compilation will terminate when the number of messages has exceeded the limit set in the MAXMSG compiler option.

IBM1723I S **Result of concatenating two literals is too long.**

Explanation: The length of the string literal produced by concatenating two string literals must not be greater than the maximum allowed for a literal with the derived string type.

IBM1724I S **Addition of** *source type* **and** *target type* **is invalid.**

Explanation: One of the operands in an addition must be computational and the other must be either computational or a locator.

IBM1725I S **Addition of** *source type* **and** *target type* **is invalid. If an ENTRY should be invoked, an argument list must be provided.**

Explanation: An ENTRY cannot be used as an arithmetic operand. If the ENTRY is a function which should be invoked, an argument list, even if it consists only of a left and right parenthesis, must be provided.

IBM1726I S **Subtraction of** *target type* **from** *source type* **is invalid.**

Explanation: The first operand in a subtraction must be computational or a locator. The second operand can be a locator only if the first is a locator. Otherwise, the second operand must be computational.

IBM1727I S **Subtraction of** *target type* **from** *source type* **is invalid. If an ENTRY should be invoked, an argument list must be provided.**

Explanation: An ENTRY cannot be used as an arithmetic operand. If the ENTRY is a function which should be invoked, an argument list, even if it consists only of a left and right parenthesis, must be provided.

IBM1728I S **Multiplication of** *source type* **by** *target type* **is invalid.**

Explanation: Both operands in a multiplication must be computational.

IBM1729I S **Multiplication of** *source type* **by** *target type* **is invalid. If an ENTRY should be invoked, an argument list must be provided.**

Explanation: An ENTRY cannot be used as an arithmetic operand. If the ENTRY is a function which should be invoked, an argument list, even if it consists only of a left and right parenthesis, must be provided.

IBM1730I S **Division of** *source type* **by** *target type* **is invalid.**

Explanation: Both operands in a division must be computational.

IBM1731I S **Division of** *source type* **by** *target type* **is invalid. If an ENTRY should be invoked, an argument list must be provided.**

Explanation: An ENTRY cannot be used as an arithmetic operand. If the ENTRY is a function which should be invoked, an argument list, even if it consists only of a left and right parenthesis, must be provided.

IBM1732I S Unsupported use of aggregate expression.

Explanation: Aggregate expressions are supported only as the source in an assignment statement and, with some limitations, as an argument to the ANY or ALL built-in functions.

IBM1733I S Concatenate operands must have computational type.

Explanation: Only expressions having string or numeric type may be concatenated.

IBM1734I S Operand in a prefix expression is not computational.

Explanation: The prefix operators (plus, minus, and logical not) may be applied only to expressions having string or numeric type.

IBM1735I S AREA variables may not be compared.

Explanation: No relational operations are defined for AREA variables.

IBM1736I S Comparison of *source type* to *target type* is invalid.

Explanation: Computational types can be compared only with other computational types, and non-computational types can be compared only with like non-computational types.

IBM1737I S Comparison of ENTRY to *target type* is invalid. If the ENTRY should be invoked, an argument list must be provided.

Explanation: ENTRYs can be compared only with other ENTRYs. If the ENTRY is a function which should be invoked, an argument list, even if it consists only of a left and right parenthesis, must be provided.

IBM1738I S Comparison of *source type* to ENTRY is invalid. If the ENTRY should be invoked, an argument list must be provided.

Explanation: ENTRYs can be compared only with other ENTRYs. If the ENTRY is a function which should be invoked, an argument list, even if it consists only of a left and right parenthesis, must be provided.

IBM1739I S TASK variables may not be compared.

Explanation: No relational operations are defined for TASK variables.

IBM1740I S Comparison of an OFFSET to a POINTER is invalid since the OFFSET comparand is not an OFFSET variable declared with an AREA qualifier.

Explanation: An OFFSET can be compared with a POINTER as long as the OFFSET can be converted to a POINTER. This requires that the OFFSET is declared with an AREA qualifier.

IBM1741I S Operands in comparison have differing strong types.

Explanation: Comparisons of strongly-typed variables are invalid unless both have the same type.

```
dcl hp  handle point;
dcl hr  handle rectangle;

if hp = hr then
  ...
```

IBM1742I S Compared ORDINALs must have the same ORDINAL type.

Explanation: ORDINALs cannot be compared with other ORDINALs having a different ORDINAL type.

IBM1743I S Source and target in assignment have differing strong types.

Explanation: Assignments of strongly-typed variables are invalid unless both have the same type.

IBM1744I S Conversion of ORDINALs is invalid unless both have the same ORDINAL type.

Explanation: ORDINALs cannot be assigned to other ORDINALs having different ORDINAL type.

IBM1745I S In a function that returns a strong type, the type in any RETURN expression must be the same as that returned by the function.

Explanation: For instance, in a function that returns a typed structure, any RETURN expression must have the same structure type.

IBM1746I S VALUE, VALUELIST, VALUERANGE, and STATIC INITIAL expressions must be constant.

Explanation: These expressions must be reducible to a constant at compile-time.

```
dcl a fixed bin static nonassignable init(0);
dcl m fixed bin value( a );
dcl n fixed bin static init( a );
```

IBM1747I S Function cannot be used before the function's descriptor list has been scanned.

Explanation: This is a compiler restriction. Reorder the declarations and blocks in your program. For example, the following declarations should be in reverse order.

```
dcl a char( csize( x, y ) );
dcl csize entry( char(2), fixed bin )
           returns( fixed bin );
```

IBM1748I S Extents of automatic variables must not depend on the extents of automatic variables declared later in the same block.

Explanation: Reorder the declarations in your program. For example, the following declarations should be in reverse order.

```
dcl a char( length(b) ) auto;
dcl b char( 10 ) auto;
```

IBM1749I S VALUE and INITIAL expressions must be scalars.

Explanation: Aggregate expressions are not valid as INITIAL and VALUE expressions.

IBM1750I S INITIAL attribute is invalid for the STATIC LABEL variable *variable-name* **since it has the MEMBER attribute.**

Explanation: The INITIAL attribute is supported for a STATIC LABEL variable only if the variable is a scalar or an array of scalars.

IBM1751I S INITIAL attribute is valid for the STATIC ENTRY variable *variable-name* **only if it has the LIMITED attribute.**

Explanation: ENTRY variables that don't have the LIMITED attribute require block activation information, and hence they cannot be initialized at compile-time.

IBM1753I S INITIAL attribute is invalid for the STATIC FORMAT variable *variable-name***.**

Explanation: FORMAT variables require block activation information, and hence they cannot be initialized at compile-time. If the variable were not a member of a structure, the storage class would be changed to AUTOMATIC and an error message would be issued instead.

IBM1754I S An asterisk iteration factor can be applied only to the last expression in the INITIAL item list for *variable-name***.**

Explanation: Since an asterisk iteration factor completes the initialization of a variable, it cannot be followed by more initial values.

```
dcl a(10)  fixed bin init( 1, 2, (*) 0, 8 );
```

IBM1755I S An asterisk iteration factor cannot be used in the nested INITIAL item list for *variable-name***.**

Explanation: An asterisk iteration can be used only in a non-nested INITIAL item list. The following example is invalid.

```
dcl a(20) fixed bin init( (2) ( 1, (*) 2 ) );
```

IBM1756I S The scalar variable *variable-name* **has an INITIAL list with more than one item.**

Explanation: Only arrays can have an INITIAL list with more than one element.

```
dcl a  fixed bin init( 1, 2 );
```

IBM1757I S LABEL constant in STATIC INITIAL for the variable *variable-name* **must be in the same block as the LABEL being initialized.**

Explanation: Change the storage class to AUTOMATIC.

```
lx:;

subproc: proc;

  dcl la  static label init( lx );

end;
```

IBM1758I S Only one element in the STATIC UNION *variable-name* **may have the INITIAL attribute.**

Explanation: If more than one element in a STATIC UNION had an INITIAL value, it would not be clear which should take precedence.

```
dcl
```

```
1 a union static,
  2 b   fixed bin(31) init( 17 ),
  2 c   fixed bin(15) init( 19 );
```

IBM1759I S Non-null INITIAL values are not supported for the STATIC NONCONNECTED array *variable-name* **since it has the attributes UNALIGNED BIT.**

Explanation: The only supported INITIAL values for a STATIC UNALIGNED BIT variable with inherited dimensions are bit strings equal to "b.

```
dcl
  1 a(10,2) static,
    2 b1  bit(1) init( (20) '1'b ),
    2 b2  bit(1) init( (20) '0'b );
```

IBM1760I S LABEL constant in the STATIC INITIAL list for *variable-name* **must not be an element of a LABEL CONSTANT array.**

Explanation: Replace the subscripted LABEL with an unsubscripted one or change the storage class to AUTOMATIC.

```
lx(1):;
lx(2):;

dcl la(2) static label init( lx(2), lx(1) );
```

IBM1761I S ENTRY reference in INITIAL clause for the STATIC ENTRY variable *variable-name* **must not be FETCHABLE.**

Explanation: The variable y in DCL x ENTRY LIMITED INIT(y) must not be FETCHABLE; y must not be used in a FETCH or RELEASE statement, and y must not have the OPTIONS(FETCHABLE) attribute.

IBM1762I S INITIAL iteration factor must have computational type.

Explanation: Iteration factors in INITIAL lists must have numeric or string types.

IBM1763I S INITIAL iteration factor must be a scalar.

Explanation: An iteration factor in an INITIAL list must not be an array, structure, or union.

IBM1764I S The BYVALUE attribute is invalid for strings of nonconstant length.

Explanation: Strings with nonconstant length must be passed and received by address.

```
a: proc( x );
  dcl x char(*) byvalue;
```

IBM1765I S Length of string with the VALUE attribute must be a constant or an asterisk.

Explanation: Named strings must have a constant length or a length determined from their VALUE.

```
dcl a fixed bin automatic;
dcl s char(a) value('variable length');
```

IBM1766I S VALUE for *variable-name* **must be evaluated before its first use.**

Explanation: Named constants must be evaluated before they are used. Reorder the declarations so that each named constant is declared before its first use.

```
dcl a char(n) static init( 'tooSoon' );
dcl n fixed bin value( 7 );
```

IBM1767I S Control variable in DO statement must not be a named constant.

Explanation: Named constants may not be used as control variables in DO loops.

```
dcl n fixed bin value( 7 );

do n = 1 to 5;
```

IBM1768I S Control variable in DO statement must have VARIABLE attribute.

Explanation: Constants may not be used as control variables in DO loops.

```
dcl ex external entry, (ev1, ev2) entry;

do ex = ev1, ev2;
```

IBM1769I S Control variable has type POINTER, but TO expression does not.

Explanation: If the control variable in a DO loop has POINTER type, the TO expression must have POINTER type. Implicit conversion from OFFSET to POINTER is not supported in this context.

IBM1770I S Control variable in loop with TO clause must have computational or locator type.

Explanation: In a DO loop with a TO clause, the control variable must have a type that allows a comparison of less than and greater than. This is possible only for computational and locator types.

IBM1771I S The *variable name* BUILTIN function may be used as a pseudovariable in a DO-loop only if the length of the pseudovariable reference is known at compile time.

Explanation: SUBSTR and UNSPEC may be used as pseudovariables in DO-loops only if their derived length is known at compile time.

IBM1772I S Source in DO loop initialization must be scalar.

Explanation: In a DO loop of the form DO a = b TO c, b must be a scalar.

IBM1773I S Control variable in DO statement must be a scalar.

Explanation: In a DO loop of the form DO x = .., x must be a scalar.

IBM1774I S Compiler restriction: control variable in DO statement must not be a BASED or CONTROLLED string or area that has non-constant extent.

Explanation: In a DO loop of the form DO x = .., if x is a string or an area, then it must have constant size or must be static, automatic, or defined.

IBM1775I S BY expression must have computational type.

Explanation: The expression in the BY clause of a DO loop must have a string or numeric type. It cannot have a locator type because it must be comparable to zero.

IBM1776I S BY expression must not be COMPLEX.

Explanation: The expression in the BY clause of a DO loop must be REAL.

```
dcl z cplx float;

do jx = 1 to 10 by z;
```

IBM1777I S TO expression must not be COMPLEX.

Explanation: The expression in the TO clause of a DO loop must be REAL

```
dcl z cplx float;

do jx = 1 to z;
```

IBM1778I S Control variable in loop with TO clause must not be COMPLEX.

Explanation: In a DO loop with a TO clause, the control variable must have a type that allows a comparison of less than and greater than. This is possible for numeric types only if the numeric type is REAL.

IBM1779I S TO expression must have computational type.

Explanation: The expression in the TO clause of a DO loop must have a string or numeric type.

IBM1780I S SIGNAL ANYCONDITION is invalid.

Explanation: ON ANYCONDITION may be used to trap conditions not otherwise trapped, but ANYCONDITION may not be signalled.

IBM1781I S And, or and exclusive-or of *source type* and *target type* is invalid.

Explanation: Bitwise operands must have a computational type.

IBM1782I S And, or and exclusive-or of *source type* and *target type* is invalid. If an ENTRY should be invoked, an argument list must be provided.

Explanation: An ENTRY cannot be used as a bitwise operand. If the ENTRY is a function which should be invoked, an argument list, even if it consists only of a left and right parenthesis, must be provided.

IBM1783I S BASED variable without an implicit qualifier must be explicitly qualified.

Explanation: A variable declared as BASED instead of as BASED(reference) must always be explicitly qualified. This is necessary even when the variable is an argument to built-in functions such as STORAGE.

IBM1784I S The ENTRY *variable-name* may not be used as a locator qualifier since it does not have the RETURNS attribute.

Explanation: Functions, but not subprocedures, can be

used as locator qualifiers (and then only if they return a locator).

IBM1785I S **The variable** *variable-name* **is used as a locator qualifier, but it is not a scalar.**

Explanation: Only scalars can be used as locator qualifiers.

IBM1786I S *BUILTIN name* **built-in may not be used as a locator qualifier.**

Explanation: The named built-in function cannot be used as a locator qualifier since it does not return a POINTER.

IBM1787I S **The ENTRY** *variable-name* **may not be used as a locator qualifier.**

Explanation: x(...)->y is invalid unless x returns a POINTER or an OFFSET declared with a qualifying AREA.

IBM1789I S **The qualifier** *variable-name* **does not have locator type.**

Explanation: Only POINTERs and OFFSETs declared with a qualifying AREA can be used as locator qualifiers.

IBM1790I S **Locator qualification is invalid for** *variable-name***.**

Explanation: Locator qualification is valid only for BASED variables.

IBM1791I S **The locator qualified reference** *reference name* **is ambiguous.**

Explanation: All references must be unambiguous.

IBM1792I S **The locator qualified reference** *reference name* **is unknown.**

Explanation: Locator qualified references must be explicitly declared. BASED variables may not be implicitly declared.

IBM1793I S **The** *variable name* **BUILTIN function may not be used as a pseudovariable in a DO-loop.**

Explanation: Only IMAG, REAL, SUBSTR and UNSPEC may be used as pseudovariables in DO loops.

IBM1794I S **Too many implicit locators are needed to resolve the qualification for a variable. Variable may be based on itself.**

Explanation: An implicitly qualified variable must require no more than 15 qualifiers to be completely qualified. If it requires more, this may indicate its qualifiers are too interdependent.

```
dcl a pointer based(b);
dcl b pointer based(a);
a = null();
```

IBM1795I S **The OFFSET variable** *variable-name* **may not be used as a locator qualifier since it was not declared with an AREA specification.**

Explanation: An OFFSET variable can be used as a locator qualifier only if it can be converted to a pointer value. This requires that the offset be declared with an AREA qualification.

IBM1796I S **Qualifier must be a scalar.**

Explanation: Arrays, structures, and unions may not be used as locator qualifiers.

IBM1797I S **BASED variables may not contain extents with nonconstant values if other extents use the REFER option.**

Explanation: The REFER option cannot be used in a BASED variable which also has an extent that is set by a non-constant expression.

IBM1798I S **Invalid scale factor in PICTURE specification.**

Explanation: The picture character F specifies a picture scaling factor for fixed-point decimal numbers. The number of digits following the V picture character, minus the integer specified with F, must be between -128 and 127.

IBM1799I S **Invalid characters in PICTURE specification.**

Explanation: The picture specification can contain only A X 9 for the Character Data, and only 9 V Z * , . / B S + - $ CR DB Y K E F < > for the Numeric Data. The characters between the insertion characters < > are not affected by this rule.

IBM1800I S **Invalid characters in the F scaling factor.**

Explanation: The picture character F specifies a picture scaling factor for fixed-point decimal numbers. The format is F(n) where n can be any signed integer between -128 and 127 inclusively.

IBM1801I S A character PICTURE string may have only A, X, or 9.

Explanation: The picture specification can contain only A, X, or 9 for the character data. Other characters are not permitted.

IBM1802I S Invalid precision in PICTURE fixed decimal precision.

Explanation: The number of digits for the precision field within a numeric data picture specification must be between one and the maximum allowed by the LIMITS(FIXEDDEC) option.

IBM1803I S Too many T, I, or R appear in the PICTURE specification.

Explanation: T, I, or R are the overpunched characters in the picture specification. Only one overpunched character can appear in the specification for a fixed point number. A floating-point specification can contain two (One in the mantissa field and one in the exponent field).

IBM1804I S PICTURE specifications in C-format items must be arithmetic.

Explanation: Character PICTURE specifications are not permitted in C-format items.

IBM1805I S Precision in numeric PICTURE must NOT be less than 1.

Explanation: The precision field within a numeric data picture specification must contain at least one digit.

IBM1806I S The precision in FIXED DECIMAL PICTURE is too big.

Explanation: The precision in the fixed decimal picture specification must not exceed that specified in the LIMITS compiler option.

IBM1807I S Precision in FLOAT DECIMAL PICTURE is too big.

Explanation: The precision in the float decimal picture specification is limited by the hardware to 18 digits.

IBM1808I S PICTURE string is empty.

Explanation: Null picture strings ("P) are invalid.

IBM1809I S Exponent in FLOAT PICTURE is too long. Exponent will be truncated to fit.

Explanation: The number of digits in the exponent of the float decimal picture specification is limited to 4.

IBM1810I S Exponent in FLOAT PICTURE has no digits.

Explanation: The exponent in the float decimal picture specification is missing. It must be entered even if it is zero.

IBM1811I S Exponent in PICTURE specification cannot contain V.

Explanation: V specifies an implicit decimal point. Therefore, it is not permitted in the exponent field.

IBM1812I S FLOAT PICTURE cannot contain CR, DB or F.

Explanation: Credit (CR), debit (DB), and scale factor (F) are only allowed in the FIXED picture specification.

IBM1813I S PICTURE specification is too long. Excess characters are truncated on the right.

Explanation: The compiler restrictions on the length of the picture specification are:

```
fixed decimal:  254
float decimal:  253
character data: 511
```

IBM1814I S PICTURE string has an invalid floating insertion character string.

Explanation: The floating insertion string is delimited by < >. Floating is done by the > character. The string can contain any character with one exception: the delimiters themselves. In order to include the characters < and > in the floating insertion string, these angle brackets must be used in an escaped format. << must be used to specify the character <, and <> must be used to specify the character >. So, for example, <aaa<<bbb<>ccc> denotes the insertion string aaa<bbb>ccc.

IBM1815I S *BUILTIN name* is a built-in subroutine. It should be used only in CALL statements and not as a function.

Explanation: Built-in subroutines cannot be used as functions - they can only be called. For instance, the following code is invalid

```
dcl pliretc builtin;

rc = pliretc( 16 );
```

IBM1816I S *keyword* **item** *variable name* **is not computational.**

Explanation: The expression must be arithmetic or string.

```
dcl x label variable;
put list( x );
```

IBM1817I S The KEYTO reference must be of type CHARACTER or GRAPHIC.

Explanation: The KEYTO reference should have the data type character or graphic. The reference can also be a variable with a non-numeric picture string specification.

IBM1818I S *I/O-option* **conflicts with previous options on the** *I/O-stmt* **statement.**

Explanation: An option on the I/O statement conflicts with prior options.

```
open file(f1) input output;
read file(f) into(x) set(p);
```

IBM1819I S The *I/O-option* **option is multiply specified on the** *I/O-stmt* **statement.**

Explanation: Each option may be specified only once.

```
read file(f1) ignore(1) ignore(2);
```

IBM1820I S Mandatory *I/O-option* **option not specified on the** *I/O-stmt* **statement.**

Explanation: A required statement element has not been specified.

```
open output;
write file(x);
```

IBM1821I S Reference for *from-into-option* **is an invalid element or aggregate type.**

Explanation: An invalid scalar or aggregate reference has been specified for the FROM or INTO clause in a record I/O statement. The example below will cause this message to be issued.

```
dcl f1 file;
read file(f1) into(f1);
```

IBM1822I S The *keyword-type* **expression must be computational.**

Explanation: The expression in a KEY or KEYFROM record I/O statement option must be computational data.

IBM1823I S SET reference must have locator type.

Explanation: In the SET clause of an ALLOCATE or LOCATE statement, the reference must have the type POINTER or OFFSET.

IBM1824I S *keyword* **expression must be scalar.**

Explanation: The expression in the named keyword clause must be scalar. This keyword clause could be an IF, UNTIL, WHILE, WHEN, KEY, KEYFROM or KEYTO clause.

```
dcl f1    file;
dcl x     char(10);
dcl z(10) char(10);
read file(f1) into(x) key(z);
```

IBM1825I S The reference in the *keyword* **clause cannot be a built-in function reference.**

Explanation: The references for the KEYTO, FROM, INTO, and SET record I/O options cannot be built-in functions. The example below will cause this message to be issued.

```
dcl f1    file;
dcl x     char(10);
read file(f1) into(hex(x));
```

IBM1826I S The reference in the *keyword* **clause cannot be a function invocation.**

Explanation: The references for the KEYTO, FROM, INTO, and SET record I/O options cannot be entry.

IBM1827I S The reference in the *keyword* **clause must have CHARACTER type.**

Explanation: The specified reference is invalid. It must be of type character. The example below will cause this message to be issued.

```
dcl p     pointer;
display ('what is your name?') reply(p);
```

IBM1828I S The reference in the *keyword* **clause must be a scalar variable.**

Explanation: The specified reference is invalid. It must be a scalar. The example below will cause this message to be issued.

```
dcl z(10) char(10);
display ('what is your name?') reply(z);
```

IBM1829I S The attributes of the argument in the *clause* clause conflict with its usage.

Explanation: The declared attributes conflict with their use in the statement.

```
dcl f file stream;
read file(f) into(x);
```

IBM1830I S *keyword* expression is not computational.

Explanation: The expression must be arithmetic or string.

```
dcl p pointer;
put list( ptradd(p,2) );
```

IBM1831I S The LOCATE reference *variable-name* is not implicitly qualified and is invalid without a SET clause.

Explanation: Provide a SET clause in the LOCATE statement.

```
dcl f file;
dcl x char(10) based;
locate x file(f1);
```

IBM1832I S SET reference must have POINTER type.

Explanation: The reference in the SET clause of a FETCH statement must have the POINTER type. OFFSET types are not supported in this context.

IBM1833I S The aggregate reference in the *from-into clause* clause must be CONNECTED.

Explanation: The specified reference in the FROM or INTO record I/O option is invalid. The reference must be connected. The example below will cause this message to be issued.

```
dcl f1 file;
dcl 1 a(3),
     2 b(4) char(4),
     2 c(4) char(4);

read file(f1) into(b);
```

IBM1834I S The expression in IGNORE must be computational.

Explanation: The specified expression in the IGNORE option of the READ statement must be computational. The example below will cause this message to be issued.

```
dcl a  area;

read file(f1) ignore(a);
```

IBM1835I S The LOCATE reference *variable-name* is not a level-1 BASED variable.

Explanation: The LOCATE reference may not be a structure member and must have the storage attribute BASED.

IBM1836I S INITIAL attribute is invalid for structures.

Explanation: The INITIAL attribute is valid only for scalars and arrays of scalars.

IBM1837I S The reference in the *keyword* clause cannot be a named constant.

Explanation: The specified reference is invalid. It cannot be a named constant. The example below will cause this message to be issued.

```
dcl f1  file;
dcl x   char(2);
dcl val fixed bin(15) value(4);

read file(f1) into(x) keyto(val);
```

IBM1838I S The attributes of *argument-number* conflict with its usage in data directed I/O.

Explanation: Only AUTOMATIC, CONTROLLED, PARAMETER, STATIC and and implicitly qualified BASED variables are supported in data directed I/O.

```
dcl q based;
put data(q);
```

IBM1839I S DATA-directed I/O does not support references with locators.

Explanation: Use a temporary or use LIST- or EDIT directed I/O.

IBM1840I S Subscripted references are not allowed in GET DATA.

Explanation: Use a temporary or use GET LIST or GET EDIT.

IBM1841I S The first argument in the *keyword*-format item is invalid.

Explanation: The format argument is outside the valid range.

```
put edit('hi') (a( -1) );
```

IBM1842I S **The field width specified in the *keyword*-format item is too small for complete input or output of the data item.**

Explanation: The width specified is too small for complete processing.

```
put edit(10190) (f(3));
```

IBM1843I S **The fractional digits specified in the *keyword*-format item is invalid.**

Explanation: The fractional number of digits must be less than or equal to the field width and non-negative.

IBM1844I S **The argument in the R-format item is not a format constant or format variable.**

Explanation: The argument to the R-format item must be either a format constant or a format variable.

IBM1845I S **The significant digits specified in E-format item is invalid.**

Explanation: The number of significant digits must be greater than or equal to the number of fractional digits, less than or equal to the field width and non-negative.

IBM1846I S **The *format-item* format item is invalid with GET/PUT STRING.**

Explanation: G, L, PAGE, LINE, SKIP, and COLUMN format items may not be used in GET/PUT EDIT statements using the STRING option.

IBM1847I S **GOTO target is inside a (different) DO loop.**

Explanation: The target of a GOTO cannot be inside a DO loop unless the GOTO itself is in the same DO loop.

IBM1848I S **The INCLUDE file for *include-stmt-arg* could not be found.**

Explanation: The INCLUDE file could not be found or opened.

IBM1849I S **Under CMPAT(V1), bounds must not be greater than 32767.**

Explanation: Under CMPAT(V1), bounds must be between -32768 and 32767 inclusive. To use bounds outside this range, specify a different CMPAT option.

IBM1850I S **Under CMPAT(V1), bounds must not be less than -32768.**

Explanation: Under CMPAT(V1), bounds must be between -32768 and 32767 inclusive. To use bounds outside this range, specify a different CMPAT option.

IBM1851I S **The INCLUDE file *include-file-name* could not be opened.**

Explanation: An unexpected error occurred while trying to open an include source file.

IBM1852I S **The preprocessor *preprocessor* is not known to the compiler.**

Explanation: A preprocessor specified in the PP compiler option is unknown.

IBM1853I S **Variable in *statement* statement must be a FETCHABLE entry constant.**

Explanation: The argument in the FETCH and RELEASE statements must be a FETCHABLE entry constant.

IBM1854I S **Fetch of the *PP name* preprocessor failed with ONCODE= *oncode*.**

Explanation: The compiler attempted to load the module specified in the PP-DEF installation option for the preprocessor.

IBM1855I S **Preprocessor *PP name* terminated abnormally with ONCODE= *oncode-value*.**

Explanation: A terminating error was detected in a preprocessor invoked by the compiler.

IBM1856I S **Fetch of the user exit initialization routine failed with ONCODE= *oncode*.**

Explanation: The compiler was unable to load the user exit.

IBM1857I S **User exit routine terminated abnormally with ONCODE= *oncode-value*.**

Explanation: The compiler detected a terminating error in the user exit.

IBM1858I S **Compilation aborted by user exit.**

Explanation: The user exit aborted the compilation by setting the return code to 16.

IBM1859I S The first statement must be a PROCEDURE or PACKAGE statement.

Explanation: All other statements must be enclosed in a PACKAGE or PROCEDURE statement.

IBM1860I S PACKAGE statement must be the first statement in the program.

Explanation: PACKAGE statements cannot follow any other statements in the program.

IBM1861I S All statements other than DECLARE, DEFAULT and PROCEDURE statements must be contained inside a PROCEDURE.

Explanation: This message can occur, for instance, if the first PROCEDURE statement is invalid or if a PROCEDURE contains too many END statements.

IBM1862I S Statements are nested too deep.

Explanation: The nesting of PROCEDURE, DO, SELECT and similar statements is greater than that supported by the compiler. Rewrite the program so that it is less complicated.

IBM1863I S Variables declared in a PACKAGE outside of any PROCEDURE must have the storage class STATIC, BASED or CONTROLLED or must be DEFINED on STATIC.

Explanation: AUTOMATIC variables must be declared inside a PROCEDURE, and DEFINED variables declared outside a PROCEDURE must be defined on STATIC.

IBM1864I S The *function name* built-in is not supported.

Explanation: Support for the indicated built-in function has been discontinued.

IBM1865I S The only BASED variables supported in data-directed i/o are those that have constant extents and that are implicitly qualified by simple variables.

Explanation: The variable implicitly qualifying the BASED variable must be a scalar that is not part of an array, structure or union, and it must be a POINTER with either the AUTOMATIC or STATIC storage attribute.

IBM1866I S The *keyword* statement is not supported.

Explanation: Support for the indicated statement has been discontinued.

IBM1867I S The pseudovariable *variable name* is not supported.

Explanation: Support for the indicated pseudovariable has been discontinued.

IBM1868I S Invalid use of iSUB.

Explanation: iSUB references are permitted only in DEFINED clauses.

IBM1869I S ALLOCATE with attribute lists is not supported.

Explanation: For example, neither of the following are supported.

```
allocate x(5);
allocate y char(10);
```

IBM1870I S ON statement cannot specify both SYSTEM and an ON-unit.

Explanation: If the SYSTEM action is specified in an ON statement, an ON-unit may not be specified as well.

```
on error system stop;
```

IBM1871I S The reference in the CONDITION condition must have type CONDITION.

Explanation: x in CONDITION(x) refers to a variable that does not have the type CONDITION.

IBM1872I S The reference in the *condition-name* condition must have type FILE.

Explanation: The reference in the named FILE condition does not have the type FILE.

IBM1873I S Nesting of DO statements exceeds the maximum.

Explanation: DO statements can be nested only 50 deep. Simplify the program.

IBM1874I S Nesting of IF statements exceeds the maximum.

Explanation: IF statements can be nested only 50 deep. Simplify the program.

IBM1875I S **Nesting of SELECT statements exceeds the maximum.**

Explanation: SELECT statements can be nested only 50 deep. Simplify the program.

IBM1876I S **Nesting of blocks exceeds the maximum.**

Explanation: Blocks may be nested only 30 deep.

IBM1878I S **The reference in the EVENT clause must have type EVENT.**

Explanation: A reference of any other type is invalid and is invalid.

IBM1879I S **The reference in the TASK clause must have type TASK.**

Explanation: A reference of any other type is invalid and is invalid.

IBM1880I S **Reference must have FILE type.**

Explanation: A file variable or constant is required.

```
dcl x format variable;
open file(x);
```

IBM1881I S **The reference** *reference name* **is ambiguous.**

Explanation: Enough qualification must be provided to make any reference unique.

IBM1882I S **The ALLOCATE reference** *variable-name* **is not a level-1 BASED or CONTROLLED variable.**

Explanation: References in ALLOCATE statements must be level-1 variable names, and those variables must have the BASED or CONTROLLED attributes.

IBM1883I S **The ALLOCATE reference** *variable-name* **is not implicitly qualified and is invalid without a SET clause.**

Explanation: Provide a SET clause in the ALLOCATE statement.

```
dcl a based;

allocate a;
```

IBM1884I S **The reference** *variable-name* **in the GENERIC attribute list is not a scalar ENTRY reference.**

Explanation: A reference of any other type is invalid.

IBM1885I S **IN option reference must have AREA type.**

Explanation: A reference of any other type is invalid.

IBM1886I S **The REFER object name** *reference name* **is ambiguous.**

Explanation: Provide enough qualification to make the name unique.

```
dcl
  1 a based,
    2 b1,
      3 c        bit(8) aligned,
      3 d        char(10),
    2 b2,
      3 c        bit(8) aligned,
      3 d        char(10),
    2 e( n refer(c)) char(10);
```

IBM1887I S **The REFER object** *reference name* **must be an element of the same structure where it is used, and must precede its first usage in that structure.**

Explanation: The named REFER object cannot be declared in another structure or in the same structure, but after its first usage.

IBM1888I S **The REFER object** *reference name* **must have computational type.**

Explanation: It must be possible to convert the REFER object safely to and from REAL FIXED BIN(31,0).

```
dcl
  1 a based,
    2 b,
      3 c        pointer,
      3 d        char(10),
    2 e( n refer(c)) char(10);
```

IBM1889I S **The REFER object** *reference name* **must be a scalar.**

Explanation: The REFER object may not have any dimensions in its declaration and neither may any of its parents.

```
dcl
  1 a based,
    2 b(8),
```

```
    3 c        fixed bin,
    3 d        char(10),
  2 e( n refer(c)) char(10);
```

IBM1890I S The REFER object *reference name* must precede the first level-2 element containing a REFER.

Explanation: Reorder the elements in the declaration so that all REFER objects precede the first level-2 element containing a REFER.

```
dcl
  1 a based,
    2 b        fixed bin,
    2 c        char( n refer(b) ),
    2 d        fixed bin,
    2 e        char( n refer(d) );
```

IBM1891I S REFER is not allowed on non-BASED variables.

Explanation: REFER can be used only in declarations of BASED variables.

IBM1892I S The REFER object *reference name* must have constant length.

Explanation: If a REFER object is a string, it must have constant length.

IBM1893I S REFER is allowed only on members of structures and unions.

Explanation: REFER cannot be used only in declarations of scalars or arrays of scalars.

IBM1894I S REINIT references must not be subscripted.

Explanation: In the statement REINIT x, x must not have any subscripts or arguments.

IBM1895I S Operations involving OPTIONS(*language-name*) routines are not supported if the DIRECTED option applies.

Explanation: If the DIRECTED(ASM) option is used, comparisons and assignments are not supported for ENTRYs declared with OPTIONS(ASM). Similarly, if the DIRECTED(COBOL) option is used, comparisons and assignments are not supported for ENTRYs declared with OPTIONS(COBOL).

IBM1896I S OPTIONS(*language-name*) is not supported for ENTRY VARIABLEs if the DIRECTED option applies.

Explanation: If the DIRECTED(ASM) option is used, ENTRY VARIABLES may not be declared with OPTIONS(ASM). Similarly, if the DIRECTED(COBOL) option is used, ENTRY VARIABLES may not be declared with OPTIONS(COBOL).

IBM1897I S Simple defining is supported only for scalars, for structures with constant extents matching those in the base variable, and for arrays of such scalars and structures as long as the array is not based on a controlled variable.

Explanation: If simple defining is not intended, specify POSITION(1) to force string defining.

IBM1898I S The base reference in the DEFINED attribute cannot be a built-in or type function.

Explanation: You can define a variable only another user variable.

IBM1899I S The base variable in the DEFINED attribute cannot be BASED, DEFINED or CONSTANT.

Explanation: Convert the DEFINED and base variables into a UNION.

IBM1900I S Extents for DEFINED bit structures must be constant.

Explanation: All bounds and string lengths for DEFINED structures and unions consisting of bit strings must be constant.

IBM1901I S POSITION attribute is invalid without the DEFINED attribute.

Explanation: The POSITION attribute has no meaning without DEFINED attribute.

IBM1902I S The expression in the POSITION attribute must have computational type.

Explanation: The POSITION expression must have a numeric or string type.

IBM1903I S The expression in the POSITION attribute for bit string-overlay defining must be an integer constant.

Explanation: The compiler must be able to evaluate the expression to an integer constant when it scans the POSITION attribute.

IBM1904I S **Variable following the *free clause* clause must be level-1 and either BASED or CONTROLLED.**

Explanation: A variable that is either based or controlled should immediately follow the FREE keyword.

IBM1905I S ***IN or SET option* option invalid after the CONTROLLED variable in the *ALLOCATE or FREE clause* clause.**

Explanation: An invalid option immediately follows a controlled variable in an ALLOCATE or FREE statement.

IBM1906I S **The reference qualifying an OFFSET attribute must be a scalar AREA reference.**

Explanation: Using the specified AREA reference to qualify an OFFSET variable is invalid. The reference must be scalar. The following example will issue this message.

```
dcl a(10) area;
dcl o     offset(a);
```

IBM1907I S **Extents for CONTROLLED variables cannot be specified using asterisks or REFER.**

Explanation: The extent specified for the controlled variable is invalid. The following example will emit this message.

```
dcl c(*)  char(10) controlled;
```

IBM1908I S **Extents for *attribute* variables cannot be specified using asterisks or REFER.**

Explanation: Extents for AUTOMATIC and DEFINED variables must be specified by expressions.

IBM1909I S **The *attribute* attribute conflicts with the *attribute* attribute.**

Explanation: The named attributes, for example PARAMETER and INITIAL, are mutually exclusive.

IBM1910I S **The attributes given in the declaration for *identifier* conflict with its use as a parameter.**

Explanation: Parameters can have no storage attributes other than CONTROLLED. Parameters also cannot have any of the attributes BUILTIN, CONDITION, CONSTANT, EXTERNAL, and GENERIC.

IBM1911I S **Repeated specifications of the unsubscripted statement label *character* are in error.**

Explanation: All statement labels in any block must be unique.

IBM1912I S **Indices specified for the LABEL *character* have already been specified.**

Explanation: All statement labels in any block must be unique.

IBM1913I S **ON-units may not be labeled. All such labels will be ignored.**

Explanation: A BEGIN block or a statement associated with an ON clause may not have a label.

IBM1914I S **GOTO target must be a LABEL reference.**

Explanation: x in GOTO x must have type LABEL. x must not have type FORMAT.

IBM1915I S **GOTO target must be a scalar.**

Explanation: x in GOTO x must not be an array.

IBM1916I S **The procedure/entry *proc-name* has already been defined.**

Explanation: Sister procedures must have different names.

```
a: proc;
  b: proc;
  end;
  b: proc;
  end;
end;
```

IBM1917I S **Program contains no valid source lines.**

Explanation: The source contains either no statements or all statements that it contains are invalid.

IBM1918I S **All the names in the ORDINAL *ordinal-name* have been previously declared.**

Explanation: None of the names in an ORDINAL should have been declared elsewhere. If they are, perhaps the ORDINAL definition has been accidentally repeated.

IBM1919I S The EXTERNAL name *string* is specified for the differing names *name* and *name*.

Explanation: Each EXTERNAL name must be used only once. So, for example, the following declares would be illegal since the external name Z is specified for two different names X and Y.

```
dcl X fixed bin(31) ext('Z');
dcl Y fixed bin(31) ext('Z');
```

IBM1920I S FIXED BINARY constant contains too many digits.

Explanation: The maximum precision of FIXED BINARY constants is set by the FIXEDBIN suboption of the LIMITS compiler option.

IBM1921I S FIXED DECIMAL constant contains too many significant digits.

Explanation: The maximum precision of FIXED DECIMAL constants is set by the FIXEDDEC suboption of the LIMITS compiler option.

IBM1922I S Exponent in FLOAT BINARY constant contains more digits than the implementation maximum.

Explanation: The exponent in a FLOAT BINARY constant may contain no more than 5 digits.

IBM1923I S Mantissa in FLOAT BINARY constant contains more significant digits than the implementation maximum.

Explanation: The mantissa in a FLOAT BINARY constant may contain no more than 64 digits.

IBM1924I S Exponent in FLOAT DECIMAL constant contains more digits than the implementation maximum.

Explanation: The exponent in a FLOAT BINARY constant may contain no more than 4 digits.

IBM1925I S Mantissa in FLOAT DECIMAL constant contains more significant digits than the implementation maximum.

Explanation: The mantissa in a FLOAT DECIMAL constant may contain no more than maximum number of digits allowed on the platform.

IBM1926I S Constants must not exceed 8192 bytes.

Explanation: The number of bytes used to represent a constant in your program must not exceed 8192. This limit holds even for bit strings where the internal representation will consume only one-eighth the number of bytes as the external representation does.

IBM1927I S SIZE condition raised by attempt to convert *source-value* to *target-attributes*

Explanation: The source value is not in the domain of the target.

```
dcl x fixed bin(15);
x = 172900;
```

IBM1928I S ERROR raised while building CEEUOPT from PLIXOPT.

Explanation: The ERROR condition was while the compiler was trying to build CEEUOPT from PLIXOPT. There may be an error in the LE APIs used by the compiler. Contact IBM service.

IBM1929I S Unable to open file *file-name* in routine *proc-name*(*line-number*).

Explanation: The compiler was unable to open the named temporary file used to communicate with the code generation module. Check the value of the TMP environment variable.

IBM1930I S Unable to write to file *file-name* . Disk may be full.

Explanation: The compiler was unable to write to a temporary file used to communicate with the code generation module. The disk to which the TMP environment variable points may be full.

IBM1932I S Unable to close file *file-name* in routine *proc-name*(*line-number*).

Explanation: The compiler was unable to close the named temporary file used to communicate with the code generation module. Check the value of the TMP environment variable.

IBM1933I S Unable to open temporary files because the path and filename are too long.

Explanation: Shorten the name of the source file or the directory specified by the TMP variable.

IBM1934I S If a parameter is a structure with nonconstant extents, only matching structures are supported as arguments.

Explanation: Assign the structure to a temporary and pass the temporary, or omit the parameter description in the entry declaration.

IBM1935I S **Structure expressions as arguments are not supported for undescribed parameters.**

Explanation: Assign the structure to a temporary and pass the temporary, or describe the parameter in the entry declaration.

IBM1936I S **Invocation of compiler backend ended abnormally.**

Explanation: The back end of the compiler either could not be found or else it detected an error from which it could not recover. The latter problem can sometimes occur, on Intel, if your disk is short of free space and, on the z/Series, if your job's region size is not large enough. Otherwise, report the problem to IBM.

IBM1937I S **Extents for parameters must be asterisks or restricted expressions with computational type.**

Explanation: For parameters, each array bound, string length and AREA size must be specified either with an asterisk or with a restricted expression that has computational type.

IBM1938I S **Message file** *file name* **not found.**

Explanation: The message must be in the current directory or in one of the directories specified in the DPATH environment variable.

IBM1939I S **Exponentiation operands must have computational type.**

Explanation: The operands in an exponentiation must have numeric or string type.

IBM1940I S *note*

Explanation: This message is used by %NOTE statements with a return code of 12.

IBM1941I U *note*

Explanation: This message is used by %NOTE statements with a return code of 16.

IBM1942I S **The scale factor specified in** *BUILTIN name* **built-in must be a restricted expression with integer type.**

Explanation: This applies to all the precision-handling built-in functions.

IBM1943I S **The number of error messages allowed by the FLAG option has been exceeded.**

Explanation: Compilation will terminate when the number of messages has exceeded the limit set in the FLAG compiler option.

IBM1944I S **The precision specified in** *BUILTIN name* **built-in must be a restricted expression with integer type.**

Explanation: This applies to all the precision-handling built-in functions.

IBM1945I S **Extents for BASED variable may not contain asterisks.**

Explanation: Extents in BASED variables must be either constants or specified with the REFER option.

IBM1946I S **Reference must be an AREA variable.**

Explanation: The specified reference is invalid. An AREA variable is needed.

IBM1947I S **The reference to the GENERIC variable** *GENERIC variable name* **cannot be resolved.**

Explanation: The argument list in a GENERIC reference must match one of the generic descriptors in one of that GENERIC's WHEN clauses. If an OTHERWISE clause was specified, the argument list must have the same number of elements as the OTHERWISE entry reference.

IBM1948I S *condition-name* **condition with ONCODE=***oncode-value* **raised while evaluating restricted expression.**

Explanation: Compile-time evaluation of a restricted expression raised a condition.

```
display( 1/0 );
```

IBM1949I S **Parameter name** *identifier* **appears more than once in parameter list.**

Explanation: Each identifier in a parameter list must be unique.

```
a: proc( b, c, b );
```

IBM1951I S *storage class* **variables must be named.**

Explanation: Variables with the CONTROLLED attribute must be named, and a variable with the EXTERNAL attribute may not have an * instead of a

name unless a name is given with the EXTERNAL attribute itself.

IBM1952I S INITIAL CALL cannot be used to initialize STATIC data.

Explanation: An INITIAL CALL must be evaluated at run-time; it can be used to initialize only non-STATIC data.

IBM1953I S The attributes of the EXTERNAL variable *variable name* **do not match those in its previous declaration.**

Explanation: EXTERNAL variables can be declared in more than one procedure in a compilation unit, but the attributes in those declarations must match.

IBM1954I S The base reference in the DEFINED attribute must be CONNECTED.

Explanation: Variables cannot be DEFINED on NONCONNECTED references.

IBM1955I S Repeated declarations of the EXTERNAL *attribute variable name* **are not supported.**

Explanation: EXTERNAL FILE constants and CONDITIONs may be declared only once in a compilation unit. Remove all but the outermost declare.

IBM1956I S ITERATE is valid only for iterative DO-groups.

Explanation: ITERATE is not valid inside type-I do groups.

IBM1957I S The WAIT event number specification must be computational.

Explanation: The expression representing the number of items to wait for in a WAIT statement is invalid. The expression must be of computational type. The following example will issue this message.

```
dcl e event;
dcl p pointer:
wait (e) (p);
```

IBM1958I S References in the WAIT statement must be of type EVENT.

Explanation: The event reference in the WAIT statement is invalid. It must be of type EVENT. The following example will issue this message.

```
dcl e entry;
wait (e);
```

IBM1959I S Invalid aggregate expression specified in WAIT statement.

Explanation: References in WAIT statements can be scalars. The only valid aggregate reference is a simple array of events. Structures, unions, and arrays of structures or unions would be flagged as errors.

IBM1960I S *type type* **type** *type name* **is not defined.**

Explanation: If ORDINAL x is used in a declaration, x must be a defined ORDINAL type.

IBM1961I S INITIAL values for *type type* **type** *type name* **must be in increasing order.**

Explanation: Any values specified in INITIAL clauses in an ORDINAL definition must be in strictly increasing order.

IBM1962I S INITIAL values for *type type* **type** *type name* **must be less than 2G.**

Explanation: ORDINAL values must fit in the range of a FIXED BIN(31) variable.

IBM1963I S *BUILTIN name* **argument must have ORDINAL type.**

Explanation: An expression contains the named built-in function with an argument that is not an ORDINAL. This message applies, for example, to the ORDINALNAME, ORDINALPRED and ORDINALSUCC built-in functions.

IBM1964I S The attributes derived from the PROCEDURE statement for the ENTRY constant *variable name* **do not match those in its explicit declaration.**

Explanation: A label on a PROCEDURE statement constitutes a declaration for an ENTRY constant with that name. That name also appears in a DECLARE statement, but the attributes in those two declarations do not match.

IBM1965I S There is more than one element named *reference name* **in the class** *structure name***.**

Explanation: All references must be unambiguous.

IBM1966I S There is no element named *reference name* **in the class** *structure name***.**

Explanation: HANDLE qualified references must be explicitly declared.

IBM1967I S **The ENTRY *variable-name* may not be used as a handle since it does not have the RETURNS attribute.**

Explanation: Functions, but not subprocedures, can be used as handles (and then only if they return a handle).

IBM1968I S **The ENTRY *variable-name* may not be used as a handle.**

Explanation: x(...)=>y is invalid unless x returns a HANDLE.

IBM1969I S **The variable *variable-name* is used as a handle, but it is not a scalar.**

Explanation: Only scalars can be used as handles.

IBM1970I S ***BUILTIN name* built-in may not be used as a handle.**

Explanation: The named built-in function cannot be used as a handle.

IBM1971I S **The GENERIC variable *variable-name* may not be used as a handle.**

Explanation: GENERIC references may not be used as handles.

IBM1972I S ***variable-name* may not be used as a handle.**

Explanation: x=>y is invalid unless x has the HANDLE attribute

IBM1976I S **DBCS characters are allowed only in G and M constants.**

Explanation: Hex strings (strings ending in one of the suffixes X, BX, B4, GX or XN), bit strings, (strings ending in the suffix B), and character strings not ending in the suffix M must contain only SBCS characters.

IBM1977I S **SBCS characters are not allowed in G constants.**

Explanation: Mixed SBCS and DBCS is allowed only in M constants.

IBM1978I S **Invalid use of SBCS encoded as DBCS.**

Explanation: Outside of comments, SBCS can be encoded as DBCS only as part of an identifier.

IBM1981I S ***BUILTIN function* may not be used outside a procedure.**

Explanation: The named built-in function may be used only inside procedures.

IBM1984I S **File *filename* could not be opened.**

Explanation: The named file could not be opened. Make sure that the file is named correctly, that it exists, that it has the proper attributes and that you have the needed permissions to access it.

IBM1985I S **File *filename* could not be opened.** *C-library-message*

Explanation: The named file could not be opened. Make sure that the file is named correctly, that it exists, that it has the proper attributes and that you have the needed permissions to access it. The accompanying C library message may help identify the problem.

IBM1986I S **A system or user abend has occurred.**

Explanation: This error can occur, for example. when writing the MDECK to a SYSPUNCH dataset that is too small or when writing to one of the other compiler output datasets when they are too small. It would probably be useful to examine the JES log.

IBM1987I S **File *filename* could not be opened because too many files have been opened.**

Explanation: The maximum number of open files has been reached. On some platforms, there is a system limit on the number of open files, but the compiler also has a limit of 2047 include files.

IBM1988I S **File *filename* could not be opened due to an access violation.**

Explanation: Either the file is in use or you tried to open a file for which you do not have sufficient privilege.

IBM1989I S **File name or extension for *filename* is too long.**

Explanation: The length of the file name or extension is greater than the maximum allowed.

IBM1990I S **File name *filename* has invalid format.**

Explanation: Apart from z/OS UNIX, file names should not contain quotes. Under z/OS UNIX, if the file name does contain quotes, it should specify a PDS member.

IBM1991I S The load of the SQL preprocessor failed with ONCODE= *oncode***. DB2/2 must be properly installed before the SQL preprocessor can be loaded.**

Explanation: The compiler attempted to load the SQL preprocessor but was unable to do so. Check that DB2/2 is properly installed.

IBM1992I S A file name must be specified.

Explanation: The command syntax is:

```
PLI {d:}{path}filename{.ext} {( options}
```

IBM1993I S Compilation terminated by ATTENTION condition.

Explanation: If you hit CTL-BRK during the compilation, the compilation will stop.

IBM1994I S Internal compiler error: storage header has been overwritten

Explanation: This message indicates that there is an error in the front end of the compiler. Please report the problem to IBM.

IBM1995I S Internal compiler error: storage tail has been overwritten.

Explanation: This message indicates that there is an error in the front end of the compiler. Please report the problem to IBM.

IBM1996I S Internal compiler error: free amount *free request size* **does not match allocated size** *allocated size***.**

Explanation: This message indicates that there is an error in the front end of the compiler. Please report the problem to IBM.

IBM1997I S Internal compiler error: no WHEN clause satisfied within *module name*

Explanation: This message indicates that there is an error in the front end of the compiler. Please report the problem to IBM.

IBM1998I S Internal compiler error: protection exception in *module name*

Explanation: This message indicates that there is an error in the front end of the compiler. Please report the problem to IBM.

IBM1999I S *note*

Explanation: This message indicates that there is an error in the back end of the compiler. Please report the problem to IBM.

IBM2000I S Internal compiler error: assertion failed on line *source line* **in** *procedure name* **in** *package name*

Explanation: This message indicates that there is an error in the front end of the compiler. Report the problem to IBM.

IBM2001I S A LICENSE REQUEST WAS DENIED FOR PL/I, PID 5655-B22. THE REQUEST ENDED WITH STATUS CODE *STATUS CODE* **AND RETURN CODE** *RETURN CODE***. THE COMPILATION WILL BE TERMINATED.**

Explanation: IBM License Manager is installed on your system, but the request to verify that you have a license to use the PL/I compiler has failed.

IBM2002I S Close of file *filename* **failed. There may be a space problem.**

Explanation: An error has occurred while attempting to close a file.

IBM2003I S Write to file *filename* **failed. There may be a space problem.**

Explanation: An error has occurred while attempting to write to a file.

IBM2004I S ATTACH reference must be declared with either a null argument list or with an argument list specifying only one argument.

Explanation: If the ATTACH reference is declared without an argument list, change the declare to specify a null argument list by adding a pair of parentheses.

IBM2005I S ATTACH reference must be an ENTRY reference.

Explanation: GENERIC references and built-in subroutines may not be attached.

IBM2006I S ATTACH reference cannot be a function reference.

Explanation: An ATTACH reference must not have the RETURNS attribute, even if the value returned is an ENTRY.

IBM2007I S ATTACH reference must use LINKAGE(SYSTEM).

Explanation: Unless the default linkage is overridden, OPTIONS(LINKAGE(SYSTEM)) must be specified on the declare for the ATTACH reference.

IBM2008I S ATTACH reference cannot be FETCHABLE.

Explanation: An ATTACH reference may not be used in a FETCH or RELEASE statement.

IBM2009I S ATTACH reference cannot be a nested procedure.

Explanation: An ATTACH reference must be a level-1 procedure, although it does need to be external.

IBM2010I S ATTACH reference, if an ENTRY variable, must be a LIMITED ENTRY.

Explanation: Specify the LIMITED attribute in the declare for the ENTRY VARIABLE.

IBM2011I S ATTACH reference, if it has an argument, must declare that argument as POINTER BYVALUE.

Explanation: No other argument types are support in ATTACH statements.

IBM2012I S The *attribute keyword* attribute is invalid in an ALIAS definition.

Explanation: The specified attribute must not be used in a DEFINE ALIAS statement. This includes attributes such as ASSIGNABLE, but, as in RETURNS descriptors, the attributes STRUCTURE, UNION and DIMENSION are not permitted in ALIAS definitions. Hence, the following are invalid:

```
define alias array (10) fixed bin;

define alias point 1, 2 fixed bin, 2 fixed bin;
```

IBM2013I S Only one description is allowed in an ALIAS definition.

Explanation: The syntax allows the name in an alias definition to be followed by a description list, but that description list must consist of exactly one description. The following is invalid:

```
define alias x fixed bin, float bin;
```

IBM2014I S Extents in type descriptors must be constant.

Explanation: In ALIAS and STRUCTURE definitions, each string length and AREA size must be specified with a restricted expression. Like RETURNS descriptors, asterisks and non-constant expressions are not permitted.

IBM2015I S VALUE attribute conflicts with data type.

Explanation: The VALUE attribute is allowed only with computational data types as well as pointer, offset, handle and ordinal.

IBM2016I S The VALUE attribute is not allowed with typed structures.

Explanation: The VALUE attribute is not allowed with typed structures.

IBM2017I S INITIAL TO is valid only for NATIVE POINTER.

Explanation: INITIAL TO is not valid for NONNATIVE POINTERs. It is also invalid for non-POINTERs since they cannot be assigned addresses.

IBM2018I S INITIAL TO is supported only for STATIC variables.

Explanation: INITIAL TO is not supported for variables belonging to any storage class other than STATIC.

IBM2019I S Unsupported LINKAGE used with the LIST attribute.

Explanation: Specify OPTIONS(LINKAGE(OPTLINK)) or, on WINDOWS, OPTIONS(LINKAGE(CDECL)) on the PROCEDURE or ENTRY having a parameter with the LIST attribute and then recompile.

IBM2020I S There is more than one element named *reference name* in the typed structure *structure name*.

Explanation: All references must be unambiguous.

IBM2021I S There is no element named *reference name* in the structure *structure name*.

Explanation: All structure references must be explicitly declared.

IBM2022I S The ENTRY *variable-name* may not be used as a typed structure qualifier since it does not have the RETURNS attribute.

Explanation: Functions, but not subprocedures, can be used as typed structure qualifiers (and then only if they return a typed structure).

IBM2023I S The ENTRY *variable-name* may not be used as a typed structure qualifier.

Explanation: x(...)=>y is invalid unless x returns a typed structure.

IBM2024I S The array variable *variable-name* may be used as a typed structure qualifier only if it is completely subscripted before its dot qualification.

Explanation: For instance, if x is an array of structure t with member m, x.m(2) is invalid. However, x(2).m is valid.

IBM2025I S *BUILTIN name* built-in may not be used as a typed structure qualifier.

Explanation: The named built-in function cannot be used as a typed structure qualifier.

IBM2026I S The GENERIC variable *variable-name* may not be used as a typed structure qualifier.

Explanation: GENERIC references may not be used as typed structure qualifiers.

IBM2027I S *variable-name* may not be used as a structure qualifier.

Explanation: x.y is invalid unless x is a structure, a union or a function returning a typed structure.

IBM2028I S TYPEs must be defined before their use.

Explanation: The DEFINE STRUCTURE or DEFINE ALIAS statement for a type x must precede any of use of x as attribute type. The following two statements should be in the opposite order.

```
dcl x type point;

define structure
  1 point,
    2 x  fixed bin(31),
    2 y  fixed bin(31);
```

IBM2029I S A DEFINE STRUCTURE statement must consist of a level one structure name optionally followed by its substructures. Use DEFINE ALIAS to set a name as a synonym for a data type.

Explanation: A DEFINE STRUCTURE statement can specify just a level 1 name only if there no other attributes specified. The following are invalid

```
define structure 1 int fixed bin;

define structure 1 a type b;
```

IBM2030I S INITIAL attribute is invalid in structure definitions.

Explanation: Defined structure types must be initialized via assignments.

IBM2031I S Storage attributes are invalid in structure definition.

Explanation: Storage attributes, such as AUTOMATIC and BYADDR, must be specified with variables declared with structure type.

IBM2032I S DEFINE STRUCTURE may not specify an array of structures.

Explanation: The level 1 name in a structure definition may not have the DIMENSION attribute.

IBM2033I S Only one description is allowed in a structure definition.

Explanation: The syntax allows the name in a structure definition to be followed by a description list, but that description list must consist of exactly one structure description. The following is invalid:

```
define structure
  1 point,
    2 x  fixed bin(31),
    2 y  fixed bin(31),
  1 rectangle,
    2 upper_left   type point,
    2 lower_right  type point;
```

IBM2034I S The argument to the type function *type function* must be an ordinal type name.

Explanation: The argument to the type functions FIRST and LAST must be a type name, and that type must be an ordinal type.

IBM2035I S **The argument to the type function *type function* must be a structure type name.**

Explanation: The argument to the type function NEW must be a type name, and that type must be a structure type.

IBM2036I S **The second argument to the type function *type function* must have locator type.**

Explanation: The second argument to the BIND type function must be a pointer or offset value that is to be converted to a handle to the structure type named as the first argument.

IBM2037I S **The first argument to the type function *type function* must be a structure type name.**

Explanation: The first argument to the type functions BIND must be a type name, and that type must be a structure type.

IBM2038I S ***BUILTIN name* argument must have HANDLE type.**

Explanation: An expression contains the named built-in function with an argument that is not a HANDLE.

IBM2039I S **Argument to *variable name* pseudovariable must be a HANDLE.**

Explanation: The TYPE pseudovariable can be applied only to HANDLEs.

IBM2040I S **The argument to the type function *type function* must be a defined type.**

Explanation: The first argument to the type function SIZE must be the name of a defined type.

IBM2041I S **The first argument to the type function *type function* must be a defined type.**

Explanation: The first argument to the type function CAST must be the name of a defined type.

IBM2042I S **The second argument to the type function *type function* must be a scalar.**

Explanation: The second argument to the type function CAST must be a scalar.

IBM2043I S **The second argument to the type function *type function* must have the same size as the first argument.**

Explanation: The second argument to the type function CAST must have the same size as the size of the type that is the first argument.

IBM2044I S **The get storage function to *BUILTIN name* must be a LIMITED ENTRY with LINKAGE(OPTLINK) and an appropriate entry description list.**

Explanation: The function should be declared as

```
dcl get entry( pointer byvalue,
               fixed bin(31) byaddr,
               fixed bin(31) byaddr )
       returns( pointer );
```

IBM2045I S **The free storage function to *BUILTIN name* must be a LIMITED ENTRY with LINKAGE(OPTLINK) and an appropriate entry description list.**

Explanation: The function should be declared as

```
dcl free entry( pointer byvalue,
                pointer byvalue,
                fixed bin(31) byvalue );
```

IBM2046I S **OPTIONS(NODESCRIPTOR) is required if the last parameter to an ENTRY or PROC has the LIST attribute.**

Explanation: If an entry or procedure has a variable number of arguments in imitation of C, i.e. if its last parameter has the LIST attribute, then OPTIONS(NODESCRIPTOR) must be specified (and valid).

IBM2047I S **The VARGLIST built-in function may be used only inside procedures whose last parameter had the LIST attribute.**

Explanation: The VARGLIST built-in function obtains the address of the variable argument list passed to procedures whose last parameter had the LIST attribute. It may not be used in subprocedures of such routines or in procedures having either no parameters or having no parameter declared with the LIST attribute.

IBM2048I S **The LIST attribute may be specified only on non-nested procedures, external entry constants, and limited entry variables.**

Explanation: The LIST attribute causes a variable argument list to be built, and such argument lists are permitted neither with nested procedures nor with

entry variables declared without the LIMITED attribute.

IBM2049I S The LIST attribute may be specified only on the last element of an entry description list.

Explanation: The LIST attribute indicates that zero or more parameters may be specified after it, but those parameters may not be described.

IBM2050I S Descriptors are supported for Fortran only for scalar character strings.

Explanation: If OPTIONS(FORTRAN DESCRIPTOR) applies, all parameters other than character strings must have constant extents.

IBM2051I S Descriptors are not supported for Fortran for routines defined by or containing ENTRY statements.

Explanation: If OPTIONS(FORTRAN DESCRIPTOR) applies to an ENTRY statement or to a procedure containing an ENTRY statement, all parameters must have constant extents.

IBM2052I S A function defined by a PROCEDURE containing ENTRY statements must return aggregate values BYADDR.

Explanation: Either BYADDR must be specified in the RETURNS option of the PROCEDURE statement, or the RETURNS(BYADDR) suboption of the DEFAULT statement must be in effect.

IBM2053I S A function defined by an ENTRY statement must return aggregate values BYADDR.

Explanation: Either BYADDR must be specified in the RETURNS option of the ENTRY statement, or the RETURNS(BYADDR) suboption of the DEFAULT statement must be in effect.

IBM2054I S A PROCEDURE containing ENTRY statements must receive all non-pointer parameters BYADDR.

Explanation: Either BYADDR must be specified in the declares for the parameters, or the BYADDR suboption of the DEFAULT statement must be in effect.

IBM2055I S An ENTRY statement must receive all parameters BYADDR.

Explanation: Either BYADDR must be specified in the declares for the parameters, or the BYADDR suboption of the DEFAULT statement must be in effect.

IBM2056I S ENTRY statement is not allowed in DO loops.

Explanation: ENTRY statements are allowed in non-iterative DO groups, but not in iterative DO loops.

IBM2057I S RETURN statement is invalid inside a BEGIN in a PROCEDURE that contains ENTRY statements.

Explanation: A RETURN statement is valid inside a BEGIN block only if the PROCEDURE enclosing that BEGIN block contains no ENTRY statements.

IBM2058I S In a PROCEDURE without the RETURNS option, any ENTRY statement must use BYADDR for its RETURNS value.

Explanation: Either BYADDR must be specified in the RETURNS option of the ENTRY statement, or the RETURNS(BYADDR) suboption of the DEFAULT statement must be in effect.

IBM2059I S OPTIONS(FORTRAN) is invalid if any parameters are UNALIGNED BIT.

Explanation: Only ALIGNED BIT strings with constant length are valid with OPTIONS(FORTRAN).

IBM2060I S Attributes may not be specified in ALLOCATEs of BASED variables.

Explanation: Attributes may be specified only in ALLOCATEs of CONTROLLED variables.

IBM2061I S Attributes specified for *variable-name* in ALLOCATE statement do not match those in its declaration.

Explanation: An attribute, such as CHARACTER, may be specified in an ALLOCATE statement only if it is also specified in the declaration of the variable to be allocated.

IBM2062I S Structuring specified in ALLOCATE of *variable-name* does not match that in its declaration.

Explanation: In an ALLOCATE statement for a structure, all the levels specified in its declaration must be specified, and no new levels may be specified.

IBM2063I S Specification of extent for *variable-name* in ALLOCATE statement is invalid since it was declared with a constant extent.

Explanation: An attribute, such as CHARACTER, may be specified in an ALLOCATE statement only if it is also specified in the declaration of the variable to be

allocated with either an asterisk or a non-constant expression.

IBM2064I S **The extent specified for the lower bound for dimension** *dimension-value* **of** *variable-name* **in ALLOCATE statement is invalid since that variable was declared with a different constant extent.**

Explanation: If a bound for a CONTROLLED variable is declared as a constant, then it must be specified as the same constant value in any ALLOCATE statement for that variable.

IBM2065I S **The extent specified for the upper bound for dimension** *dimension-value* **of** *variable-name* **in ALLOCATE statement is invalid since that variable was declared with a different constant extent.**

Explanation: If a bound for a CONTROLLED variable is declared as a constant, then it must be specified as the same constant value in any ALLOCATE statement for that variable.

IBM2075I S **ENTRY types and arguments in** *type function* **must be LIMITED.**

Explanation: A ENTRY type or argument used with the type function CAST must have the attribute LIMITED.

IBM2076I S **FLOAT types and arguments in** *type function* **must be NATIVE REAL.**

Explanation: A FLOAT type or argument used with the type function CAST must have the attributes NATIVE REAL.

IBM2077I S **FIXED BIN types and arguments in** *type function* **must be REAL with scale factor zero.**

Explanation: A FIXED BIN type or argument used with the type function CAST must have the attributes REAL PRECISION(p,0).

IBM2078I S **Types with the attributes** *attributes* **are not supported as the target of the** *type function* **function.**

Explanation: The first argument to the type function CAST must be a type with one of the following sets of attributes: REAL FIXED BIN(p,0) or NATIVE REAL FLOAT.

IBM2079I S **Arguments with the attributes** *attributes* **are not supported as the source in the** *type function* **function.**

Explanation: The second argument to the type function CAST must have one of the following sets of attributes: REAL FIXED BIN(p,0) or NATIVE REAL FLOAT.

IBM2080I S **DATE pattern is invalid.**

Explanation: See the Language Reference Manual for a list of the supported DATE patterns.

IBM2081I S **DATE attribute is valid only with NONVARYING CHARACTER, FIXED DECIMAL and arithmetic PICTURE.**

Explanation: The DATE attribute cannot be used on any other than the named types.

IBM2082I S **DATE attribute conflicts with non-zero scale factor.**

Explanation: The DATE attribute can be used on a numeric only if it has a scale factor of zero.

IBM2083I S **DATE attribute conflicts with COMPLEX attribute.**

Explanation: The DATE attribute can be used on a numeric only if it is REAL.

IBM2084I S **DATE attribute conflicts with PICTURE string containing characters other than 9.**

Explanation: The DATE attribute can be used on a PICTURE only if the PICTURE consists entirely of 9's.

IBM2085I S **Length of DATE pattern and base precision do not match.**

Explanation: The DATE attribute can be used on a numeric only if its precision equals the length of the DATE pattern.

IBM2086I S **Length of DATE pattern and base length do not match.**

Explanation: The DATE attribute can be used on a string only if its length equals the length of the DATE pattern.

IBM2087I S **DATE attribute conflicts with adjustable length.**

Explanation: The DATE attribute can be used on a string only if the string is declared with a constant length.

IBM2088I S Response file is too large. Excess will be ignored.

Explanation: The options string built from the response file must be less than 32767 characters long.

IBM2089I S Line in response file is longer than 100 characters. That line and rest of file will be ignored.

Explanation: All lines in any response file must contain no more than 100 characters.

IBM2090I S The *keyword* statement cannot be used under SYSTEM(CICS).

Explanation: The named statement cannot be used under CICS.

IBM2091I S DISPLAY with REPLY cannot be used under SYSTEM(CICS).

Explanation: DISPLAY with REPLY cannot be used under CICS.

IBM2092I S The *BUILTIN name* built-in function cannot be used under SYSTEM(CICS).

Explanation: The named built-in function cannot be used under CICS.

IBM2093I S The *keyword* statement cannot be used under SYSTEM(CICS) except with SYSPRINT.

Explanation: The named I/O statement cannot be used under CICS unless the file used in the statement is SYSPRINT.

IBM2094I S Source in CAST to FLOAT must be FLOAT, FIXED or ORDINAL.

Explanation: The source in a CAST to a FLOAT must be FLOAT, FIXED or ORDINAL.

IBM2095I S Target in CAST from FLOAT must be FLOAT, FIXED BIN or ORDINAL.

Explanation: The target in a CAST from a FLOAT must be FLOAT, FIXED BIN or ORDINAL.

IBM2096I S Target in CAST from FIXED DEC must be FLOAT, FIXED BIN or ORDINAL.

Explanation: The target in a CAST from a FIXED DEC must be FLOAT, FIXED BIN or ORDINAL.

IBM2097I S FIXED DEC types and arguments in *type function* must be REAL with non-negative scale factor.

Explanation: A FIXED DEC type or argument used with the type function CAST must have the attributes REAL PRECISION(p,q) with p >= q and q >= 0.

IBM2098I S Source in CAST to FIXED DEC must be FLOAT, FIXED or ORDINAL.

Explanation: The source in a CAST to a FIXED DEC must be FLOAT, FIXED or ORDINAL.

IBM2099I S CASEX strings must have the same length.

Explanation: The two strings in the CASEX option must have the same length. The second argument is the uppercase value of the first. If a character in the first string does not have an uppercase value, use the character itself as the uppercase value.

IBM2100I S The attributes of the EXTERNAL variable *variable name* do not match those in its previous declaration. The ORDINAL types do not match.

Explanation: EXTERNAL variables can be declared in more than one procedure in a compilation unit, but the attributes in those declarations must match.

IBM2101I S The attributes of the EXTERNAL variable *variable name* do not match those in its previous declaration. The HANDLE types do not match.

Explanation: EXTERNAL variables can be declared in more than one procedure in a compilation unit, but the attributes in those declarations must match.

IBM2102I S The attributes of the EXTERNAL variable *variable name* do not match those in its previous declaration. The STRUCTURE types do not match.

Explanation: EXTERNAL variables can be declared in more than one procedure in a compilation unit, but the attributes in those declarations must match.

IBM2103I S The attributes of the EXTERNAL variable *variable name* do not match those in its previous declaration. Alignment does not match.

Explanation: EXTERNAL variables can be declared in more than one procedure in a compilation unit, but the attributes in those declarations must match.

IBM2104I S **The attributes of the EXTERNAL variable *variable name* do not match those in its previous declaration. Number and attributes of structure members do not match.**

Explanation: EXTERNAL variables can be declared in more than one procedure in a compilation unit, but the attributes in those declarations must match.

IBM2105I S **The attributes of the EXTERNAL variable *variable name* do not match those in its previous declaration. The number of dimensions do not match.**

Explanation: EXTERNAL variables can be declared in more than one procedure in a compilation unit, but the attributes in those declarations must match.

IBM2106I S **The attributes of the EXTERNAL variable *variable name* do not match those in its previous declaration. Lower bounds do not match.**

Explanation: EXTERNAL variables can be declared in more than one procedure in a compilation unit, but the attributes in those declarations must match.

IBM2107I S **The attributes of the EXTERNAL variable *variable name* do not match those in its previous declaration. Upper bounds do not match.**

Explanation: EXTERNAL variables can be declared in more than one procedure in a compilation unit, but the attributes in those declarations must match.

IBM2108I S **The attributes of the EXTERNAL variable *variable name* do not match those in its previous declaration. RETURNS attributes do not match.**

Explanation: EXTERNAL variables can be declared in more than one procedure in a compilation unit, but the attributes in those declarations must match.

IBM2109I S **The attributes of the EXTERNAL variable *variable name* do not match those in its previous declaration. BYVALUE and BYADDR attributes in RETURNS do not match.**

Explanation: EXTERNAL variables can be declared in more than one procedure in a compilation unit, but the attributes in those declarations must match.

IBM2110I S **The attributes of the EXTERNAL variable *variable name* do not match those in its previous declaration. LINKAGE values do not match.**

Explanation: EXTERNAL variables can be declared in more than one procedure in a compilation unit, but the attributes in those declarations must match.

IBM2111I S **The attributes of the EXTERNAL variable *variable name* do not match those in its previous declaration. OPTIONS values do not match.**

Explanation: EXTERNAL variables can be declared in more than one procedure in a compilation unit, but the attributes in those declarations must match.

IBM2112I S **The attributes of the EXTERNAL variable *variable name* do not match those in its previous declaration. Parameter counts do not match.**

Explanation: EXTERNAL variables can be declared in more than one procedure in a compilation unit, but the attributes in those declarations must match.

IBM2113I S **The attributes of the EXTERNAL variable *variable name* do not match those in its previous declaration. BYVALUE and BYADDR attributes in parameter *parameter-number* do not match.**

Explanation: EXTERNAL variables can be declared in more than one procedure in a compilation unit, but the attributes in those declarations must match.

IBM2114I S **The attributes of the EXTERNAL variable *variable name* do not match those in its previous declaration. The number of dimensions for parameter *parameter-number* do not match.**

Explanation: EXTERNAL variables can be declared in more than one procedure in a compilation unit, but the attributes in those declarations must match.

IBM2115I S **The attributes of the EXTERNAL variable *variable name* do not match those in its previous declaration. Lower bounds for parameter *parameter-number* do not match.**

Explanation: EXTERNAL variables can be declared in more than one procedure in a compilation unit, but the attributes in those declarations must match.

IBM2116I S **The attributes of the EXTERNAL variable *variable name* do not match those in its previous declaration. Upper bounds for parameter *parameter-number* do not match.**

Explanation: EXTERNAL variables can be declared in more than one procedure in a compilation unit, but the attributes in those declarations must match.

IBM2117I S **The attributes of the EXTERNAL variable *variable name* do not match those in its previous declaration. Alignment of parameter *parameter-number* does not match.**

Explanation: EXTERNAL variables can be declared in more than one procedure in a compilation unit, but the attributes in those declarations must match.

IBM2118I S **The attributes of the EXTERNAL variable *variable name* do not match those in its previous declaration. Number and attributes of structure members in parameter *parameter-number* do not match.**

Explanation: EXTERNAL variables can be declared in more than one procedure in a compilation unit, but the attributes in those declarations must match.

IBM2119I S **The attributes of the EXTERNAL variable *variable name* do not match those in its previous declaration. Attributes of parameter *parameter-number* do not match.**

Explanation: EXTERNAL variables can be declared in more than one procedure in a compilation unit, but the attributes in those declarations must match.

IBM2120I S **AREAs are not supported in RETURNS.**

Explanation: But an AREA may be output parameter.

IBM2121I S **Argument number *argument-number* in entry reference *entry name* must have the same size as the corresponding parameter.**

Explanation: For a AREA parameter declared with constant size, any corresponding argument must have equal constant size. Dummy AREA arguments are not supported in this scenario.

```
dcl x entry( area(10000) );
dcl a area(8000) );
call x( a );
```

IBM2127I S **The ENTRY named *ENTRY variable name* matches the reference to the GENERIC variable *GENERIC variable name*, but while the GENERIC reference is used as a function, the matching ENTRY does not have the RETURNS attribute.**

Explanation: A match for the GENERIC reference has been found, but the match is not suitable because while the GENERIC reference is used as a function, the matching ENTRY is not a function. For example, the first GENERIC reference below is invalid, while the second is ok.

```
dcl e1 entry( fixed bin );
dcl e2 entry( fixed bin, fixed bin ) returns( fixed bin );
dcl gp generic( e1 when( * ),
                e2 when( *, * ) );

rc = gp( 0 );

rc = gp( 0, 0 );
```

IBM2128I S **The ENTRY named *ENTRY variable name* matches the reference to the GENERIC variable *GENERIC variable name*, but while the GENERIC reference is used as a function acting as a locator qualifier, the matching ENTRY does not return a POINTER.**

Explanation: A match for the GENERIC reference has been found, but the match is not suitable because while the GENERIC reference is used as a locator, the matching ENTRY is not a function returning a POINTER. For example, the first GENERIC reference below is invalid, while the second is ok.

```
dcl f1 entry( fixed bin ) returns( fixed bin );
dcl f2 entry( fixed bin, fixed bin ) returns( pointer );
dcl bx based fixed bin;
dcl gf generic( f1 when( * ),
                f2 when( *, * ) );

rc = gf( 0 )->bx;

rc = gf( 0, 0 )->bx;
```

IBM2129I S **The ENTRY named *ENTRY variable name* matches the reference to the GENERIC variable *GENERIC variable name*, but while the GENERIC reference is used as a repeating function reference, the matching ENTRY cannot be so used.**

Explanation: A match for the GENERIC reference has been found, but the match is not suitable because while the GENERIC reference is used as a function whose return value is a function that is invoked (and so on, as the number of argument lists mandates), the matching

ENTRY cannot be so used. For example, the first GENERIC reference below is invalid, while the second is ok.

```
dcl x1 entry( fixed bin )
       returns( entry );
dcl x2 entry( fixed bin, fixed bin )
       returns( entry returns( fixed bin ) );
dcl gx generic( x1 when( * ),
                x2 when( *, * ) );

rc = gx( 0 )();

rc = gx( 0, 0 )();
```

IBM2130I S iSUB defining is not valid with the POSITION attribute.

Explanation: The POSITION attribute can be used only with string overlay defining.

```
dcl b(4) char(2) pos(2) def( a(1sub,1sub) );
```

IBM2131I S In iSUB defining, the base and DEFINED variables must match.

Explanation: The defined and base arrays in iSUB defining must have identical attributes apart from the dimension attribute.

```
dcl a(4) fixed bin(31);
dcl b(4) fixed bin(15) def( a(1sub,1sub) );
```

IBM2132I S The i in an iSUB reference must not exceed the dimensionality of the DEFINED variable.

Explanation: The i in an iSUB reference must refer to a subscript of the DEFINED variable and hence must not be greater than the number of dimensions for that variable.

```
dcl a(4,4) fixed bin(31);
dcl b(4) fixed bin(15) def( a(1sub,2sub) );
```

IBM2133I S An iSUB variable cannot be defined on a cross-section of its base.

Explanation: In an iSUB variable, no asterisks may appear in the specification of the base array.

```
dcl a(4,4) fixed bin(31);
dcl b(4) fixed bin(15) def( a(1sub,*) );
```

IBM2134I S iSUB defining is supported only for arrays of scalars.

Explanation: iSUB defining is not supported for structures and unions.

IBM2135I S DFT(DESCLIST) conflicts with CMPAT(*cmpat-suboption*).

Explanation: If CMPAT(V1) or CMPAT(V2) is specified, then DFT(DESCLOCATOR) must be in effect (as it is by default on z/OS).

IBM2136I S The number of indices specified for the LABEL *identifier* does not match the number previously specified.

Explanation: The number of indices given for an element of a label constant array must not vary.

```
a(1,1): ....
a(1,2): ....
a(3): ....
```

IBM2137I S Indices have been specified for the LABEL *identifier* when it was previously specified without indices.

Explanation: A label constant cannot be subscripted if its first use contains no subscripts.

```
a: ....
a(3): ....
```

IBM2138I S Indices have not been specified for the LABEL *identifier* when it was previously specified with indices.

Explanation: A label constant must be subscripted if its first use contains subscripts.

```
a(3): ....
a: ....
```

IBM2139I S The Language Enviroment run-time is not current enough.

Explanation: The compiler requires that you use z/OS Language Environment V2 R1 or later.

IBM2140I S Length of second argument to the REPLACEBY2 built-in must be twice that of the third.

Explanation: The second argument to the REPLACEBY2 built-in function provides the set of pairs of characters which are to replace the corresponding characters in the third argument, and hence the length

of the second string must be twice that of the third.

IBM2141I S **First argument to the** *BUILTIN name* **built-in must be a structure.**

Explanation: The first argument to the named built-in subroutine must be a structure.

IBM2142I S **Event structure argument to the** *BUILTIN name* **built-in has too few elements.**

Explanation: The first argument to the named built-in subroutine must be a structure supplying the event handlers for the SAX parser, and that structure must have exactly the right number of members. See the Programming Guide for more details.

IBM2143I S **Event structure argument to the** *BUILTIN name* **built-in has too many elements.**

Explanation: The first argument to the named built-in subroutine must be a structure supplying the event handlers for the SAX parser, and that structure must have exactly the right number of members. See the Programming Guide for more details.

IBM2144I S **Member** *member-number* **in the event structure argument to the** *BUILTIN name* **built-in is not a scalar.**

Explanation: The first argument to the named built-in subroutine must be a structure supplying the event handlers for the SAX parser, and each element of that structure must be a scalar. See the Programming Guide for more details.

IBM2145I S **Member** *member-number* **in the event structure argument to the** *BUILTIN name* **built-in must be a LIMITED ENTRY.**

Explanation: The indicated element of the structure supplying the event handlers for the SAX parser must be a LIMITED ENTRY. See the Programming Guide for more details.

IBM2146I S **Member** *member-number* **in the event structure argument to the** *BUILTIN name* **built-in must return BYVALUE a NATIVE FIXED BIN(31).**

Explanation: The indicated element of the structure supplying the event handlers for the SAX parser must be a function returning BYVALUE a NATIVE FIXED BIN(31). See the Programming Guide for more details.

IBM2147I S **Member** *member-number* **in the event structure argument to the** *BUILTIN name* **built-in must have a non-empty entry description list.**

Explanation: The indicated element of the structure supplying the event handlers for the SAX parser must have a non-empty entry description list. See the Programming Guide for more details.

IBM2148I S **Member** *member-number* **in the event structure argument to the** *BUILTIN name* **built-in has a parameter count of** *specified-parm-count* **when the correct parameter count is** *required-parm-count* **.**

Explanation: The indicated element of the structure supplying the event handlers for the SAX parser must have the correct number of parameters. See the Programming Guide for more details.

IBM2149I S **Member** *member-number* **in the event structure argument to the** *BUILTIN name* **built-in must have a BYVALUE POINTER as its first parameter.**

Explanation: The indicated element of the structure supplying the event handlers for the SAX parser must have a BYVALUE POINTER as its first parameter. See the Programming Guide for more details.

IBM2150I S **Member** *member-number* **in the event structure argument to the** *BUILTIN name* **built-in must have a BYVALUE POINTER as its second parameter.**

Explanation: The indicated element of the structure supplying the event handlers for the SAX parser must have a BYVALUE POINTER as its second parameter. See the Programming Guide for more details.

IBM2151I S **Member** *member-number* **in the event structure argument to the** *BUILTIN name* **built-in must have a BYVALUE NATIVE FIXED BIN(31) as its third parameter.**

Explanation: The indicated element of the structure supplying the event handlers for the SAX parser must have a BYVALUE NATIVE FIXED BIN(31) as its third parameter. See the Programming Guide for more details.

IBM2152I S **Member** *member-number* **in the event structure argument to the** *BUILTIN name* **built-in must have a BYVALUE POINTER as its fourth parameter.**

Explanation: The indicated element of the structure supplying the event handlers for the SAX parser must have a BYVALUE POINTER as its fourth parameter. See the Programming Guide for more details.

IBM2153I S **Member** *member-number* **in the event structure argument to the** *BUILTIN name* **built-in must have a BYVALUE NATIVE FIXED BIN(31) as its fifth parameter.**

Explanation: The indicated element of the structure supplying the event handlers for the SAX parser must have a BYVALUE NATIVE FIXED BIN(31) as its fifth parameter. See the Programming Guide for more details.

IBM2154I S **Member** *member-number* **in the event structure argument to the** *BUILTIN name* **built-in must have a BYVALUE POINTER as its second parameter.**

Explanation: The indicated element of the structure supplying the event handlers for the SAX parser must have a BYVALUE POINTER as its second parameter. See the Programming Guide for more details.

IBM2155I S **Member** *member-number* **in the event structure argument to the** *BUILTIN name* **built-in must have a BYVALUE NATIVE FIXED BIN(31) as its fourth parameter.**

Explanation: The indicated element of the structure supplying the event handlers for the SAX parser must have a BYVALUE NATIVE FIXED BIN(31) as its fourth parameter. See the Programming Guide for more details.

IBM2156I S **Member** *member-number* **in the event structure argument to the** *BUILTIN name* **built-in must have a BYVALUE NATIVE FIXED BIN(31) as its second parameter.**

Explanation: The indicated element of the structure supplying the event handlers for the SAX parser must have a BYVALUE NATIVE FIXED BIN(31) as its second parameter. See the Programming Guide for more details.

IBM2157I S **Member** *member-number* **in the event structure argument to the** *BUILTIN name* **built-in must have a BYVALUE CHAR(1) or BYVALUE WCHAR(1) as its second parameter.**

Explanation: The indicated element of the structure supplying the event handlers for the SAX parser must have a BYVALUE CHAR (or BYVALUE WIDECHAR) of length one as its second parameter. See the Programming Guide for more details.

IBM2158I S **Member** *member-number* **in the event structure argument to the** *BUILTIN name* **built-in has the wrong linkage.**

Explanation: The indicated element of the structure supplying the event handlers for the SAX parser must have the PL/I default linkage. See the Programming Guide for more details.

IBM2159I S **Member** *member-number* **in the event structure argument to the** *BUILTIN name* **built-in must have the NODESCRIPTOR option.**

Explanation: The indicated element of the structure supplying the event handlers for the SAX parser must have the NODESCRIPTOR option. See the Programming Guide for more details.

IBM2160I S **All members of the input structure to the** *BUILTIN name* **built-in must have computational type.**

Explanation: The XMLCHAR built-in function cannot be applied to structures containing noncomputational types.

IBM2161I S **The input structure to the** *BUILTIN name* **built-in must not be a UNION or contain any UNIONs.**

Explanation: The XMLCHAR built-in function cannot be applied to unions or to structures containing unions.

IBM2162I S **The input structure to the** *BUILTIN name* **built-in must not contain any GRAPHIC elements.**

Explanation: The XMLCHAR built-in function cannot be applied to structures containing any GRAPHIC data.

IBM2163I S **The input structure to the** *BUILTIN name* **built-in must not contain any UTF-16 elements.**

Explanation: The XMLCHAR built-in function cannot be applied to structures containing any WIDECHAR or WIDEPIC data.

IBM2164I S **The input structure to the** *BUILTIN name* **built-in must not contain any unnamed substructures.**

Explanation: The XMLCHAR built-in function cannot be applied to structures containing substructures using an asterisk as a name.

IBM2165I S **PRV support is provided only if the LIMITS(EXTNAME(7)) option is in effect.**

Explanation: Support for long external names is incompatible with support for using the PRV to address CONTROLLED variables.

IBM2166I S PRV support is provided only if the NORENT option is in effect.

Explanation: Support for the RENT option is incompatible with support for using the PRV to address CONTROLLED variables.

IBM2167I S PRV support is provided only if the CMPAT(V2) or CMPAT(V3) option is in effect.

Explanation: Support for the CMPAT(LE) option is incompatible with support for using the PRV to address CONTROLLED variables.

IBM2170I S Too many INTERNAL CONTROLLED variables.

Explanation: When using the PRV to address CONTROLLED variables, there may be no more than 568 INTERNAL CONTROLLED variables.

IBM2171I S Under the NOWRITABLE option, no FETCHABLE ENTRY may be declared at the PACKAGE level.

Explanation: Under the NOWRITABLE option, every FETCHABLE ENTRY constant must be declared inside a PROCEDURE.

IBM2172I S Under the NOWRITABLE option, no FILE CONSTANT may be declared at the PACKAGE level.

Explanation: Under the NOWRITABLE option, every FILE CONSTANT must be declared inside a PROCEDURE.

IBM2173I S Under the NOWRITABLE option, no CONTROLLED may be declared at the PACKAGE level.

Explanation: Under the NOWRITABLE option, every CONTROLLED variable must be declared inside a PROCEDURE.

IBM2174I S Result of REPLACEBY2 is too long.

Explanation: The length of the string literal produced by applying the REPLACEBY2 built-in function to 3 literals must not be greater than the maximum allowed for a character literal.

IBM2175I S The second and third arguments to REPLACEBY2 must be restricted expressions.

Explanation: The REPLACEBY2 built-in function currently supports only second and third arguments that have a length and value known at compile time.

IBM2176I S The result of the *BUILTIN name* built-in would require more than 32767 bytes.

Explanation: The HEX and HEXIMAGE built-in functions cannot be applied to strings using more than 16383 bytes of storage.

IBM2177I S The file *filename* is a PDS member and hence cannot be used for SYSADATA.

Explanation: The named file is the file intended to be used as the SYSADATA file, but such a file must not be a member of a PDS.

IBM2178I S INCLUDE statements are not supported when the LINEDIR option is in effect.

Explanation: When the LINEDIR option is in effect, your source must contain no INCLUDE statements.

IBM2179I S There is too little room between the margins for the LINE directive. The PPTRACE option will be turned off.

Explanation: The %LINE directive generated by the PPTRACE must fit on one line. You must either make the margins wide enough to allow this or make the source file names short enough.

IBM2180I S Use of the KEYED DIRECT file *filename* in a *keyword* statement without a KEY/KEYFROM clause is invalid.

Explanation: Any input/output operation using a KEYED DIRECT file must include the key of the record to which the the operation is to be applied.

IBM2181I S First argument to *BUILTIN name* built-in must have type CHARACTER.

Explanation: This applies to the PICSPEC built-in function, for example.

IBM2182I S Argument number *argument number* to *BUILTIN name* built-in must be a constant.

Explanation: The specified argument to the named built-in function must be a restricted expression. This applies to second argument to the PICSPEC built-in function, for example.

IBM2183I S The first argument to *BUILTIN name* built-in must have constant length equal to that of the second argument.

Explanation: This applies to the PICSPEC built-in function, for example.

IBM2184I S **Compiler input files must have less then 1000000 lines.**

Explanation: Break up the source files into smaller files.

IBM2185I S **Argument to** *BUILTIN name* **built-in must have type REAL DECIMAL FLOAT, and the DFP option must be in effect.**

Explanation: This applies to the ISFINITE and similar built-in functions.

IBM2186I S *BUILTIN name* **is not supported for DFP.**

Explanation: The named built-in function is not supported for float using DFP. This message applies, for instance, to the SQRTF built-in functions

IBM2187I S **The exponent in the literal** *value* **is too large for DECIMAL FLOAT with precision** *precision***.**

Explanation: A DFP literal value when adjusted to have no decimal point (e.g. 3.14E0 would be adjusted to 314E-2) must have an exponent no larger than the maximum for its precision. For precision <= 7, the maximum is 90. For 7 < precision <= 16, the maximum is 369. For 16 < precision, the maximum is 6111.

IBM2188I S **The exponent in the literal** *value* **is too small for DECIMAL FLOAT with precision** *precision***.**

Explanation: A DFP literal value when adjusted to have no decimal point (e.g. 3.14E0 would be adjusted to 314E-2) must have an exponent no smaller than the minimum for its precision. For precision <= 7, the minimum is -95. For 7 < precision <= 16, the minimum is -383. For 16 < precision, the minimum is -6143.

IBM2189I S **Under CMPAT(V2) and CMPAT(LE), bounds must not be greater than +2147483647.**

Explanation: Under CMPAT(V2) and CMPAT(LE), bounds must be between -2147483648 and +2147483647.

IBM2190I S **Under CMPAT(V2) and CMPAT(LE), bounds must not be less than -2147483648.**

Explanation: Under CMPAT(V2) and CMPAT(LE), bounds must be between -2147483648 and +2147483647.

IBM2191I S **No valid character specified in the** *option* **option.**

Explanation: You must specify at least one valid character in each of the OR, NOT and QUOTE or NAMES compiler options.

IBM2192I S **Member** *member-number* **in the event structure argument to the** *BUILTIN name* **built-in must have a BYVALUE POINTER as parameter number** *parameter-number* **.**

Explanation: The indicated element of the structure supplying the event handlers for the SAX parser must have a BYVALUE POINTER in the specified parameter position. See the Programming Guide for more details.

IBM2193I S **Member** *member-number* **in the event structure argument to the** *BUILTIN name* **built-in must have a BYVALUE NATIVE FIXED BIN(31) as parameter number** *parameter-number* **.**

Explanation: The indicated element of the structure supplying the event handlers for the SAX parser must have a BYVALUE NATIVE FIXED BIN(31) in the specified parameter position. See the Programming Guide for more details.

IBM2194I S **Member** *member-number* **in the event structure argument to the** *BUILTIN name* **built-in must have a BYADDR POINTER as parameter number** *parameter-number* **.**

Explanation: The indicated element of the structure supplying the event handlers for the SAX parser must have a BYADDR POINTER in the specified parameter position. See the Programming Guide for more details.

IBM2195I S **Member** *member-number* **in the event structure argument to the** *BUILTIN name* **built-in must have a BYADDR NATIVE FIXED BIN(31) as parameter number** *parameter-number***.**

Explanation: The indicated element of the structure supplying the event handlers for the SAX parser must have a BYADDR NATIVE FIXED BIN(31) in the specified parameter position. See the Programming Guide for more details.

IBM2196I S **Member** *member-number* **in the event structure argument to the** *BUILTIN name* **built-in must have a BYVALUE ALIGNED BIT(8) as parameter number** *parameter-number* **.**

Explanation: The indicated element of the structure supplying the event handlers for the SAX parser must

have a BYVALUE ALIGNED BIT(8) in the specified parameter position. See the Programming Guide for more details.

IBM2197I S **Argument to *BUILTIN name* built-in must have type CHAR or WIDECHAR.**

Explanation: This applies to the UVALID and ULENGTH built-in functions, for example.

IBM2198I S **First argument to *BUILTIN name* built-in must have type CHAR or WIDECHAR.**

Explanation: This applies to the UPOS and UWIDTH built-in functions, for example.

IBM2199I S **The run-time option XPLINK(ON) must be in effect if object code is to be generated.**

Explanation: The compiler backend requires the XPLINK(ON) option to be in effect.

IBM2200I S **DFP conversion from *source type* to *target type* failed with an operation exception. The most likely cause for this is lack of DFP hardware.**

Explanation: The indicated conversion had a DFP source, target, or both but failed at compile time with an operation exception. These conversions require that the machine on which the compilation occurs have DFP hardware installed.

IBM2201I S **First argument to *BUILTIN name* built-in must have type REAL DECIMAL FIXED, or REAL DECIMAL FLOAT, and in the latter case, the DFP option must be in effect.**

Explanation: This applies to the ROUNDDEC and similar built-in functions.

IBM2202I S **Use of the *BUILTIN name* built-in requires ARCH(*level*) or greater.**

Explanation: This applies to various built-in functions on some platforms. For example, on z/OS, MEMCU4* and MEMCU*4 require at least ARCH(7).

IBM2203I S **The VALUE attribute may be used on a structure member only if it is used on all base members of that structure.**

Explanation: If any leaf structure member has the VALUE attribute, then all must have the VALUE attribute.

IBM2204I S **The VALUE attribute may be used on a structure member only if no storage attribute is specified for the structure.**

Explanation: It is invalid to specify the VALUE attribute for a member of a structure if the structure has a storage attribute such as BASED, CONTROLLED, etc.

IBM2205I S **The VALUE attribute may be used on a structure member only if no dimension attributes are specified for its parents.**

Explanation: It is invalid to specify the VALUE attribute for a member of a structure that has inherited dimensions.

IBM2206I S **The VALUE attribute conflicts with the DIMENSION attribute.**

Explanation: It is invalid to specify the VALUE attribute for an array.

IBM2207I S **The VALUE attribute may be used on a structure member only if no parent has the UNION attribute.**

Explanation: It is invalid to specify the VALUE attribute for a member of a union.

IBM2208I S **References to a structure containing elements with the VALUE attribute are invalid.**

Explanation: Only the leaf elements of such a structure may be referenced.

IBM2209I S **Use of nonconstant extents in BASED variables without REFER is invalid except on scalars.**

Explanation: Extents in BASED variables must all be constant except where the REFER option is used - unless the variable is a scalar. So, the first declare below is valid, while the second is invalid.

```
dcl x based char(n);
dcl y(n,m) based fixed bin(31);
```

IBM2210I S **The VALUE type function cannot be applied to *type name* since that structure has no members with an INITIAL attribute.**

Explanation: The VALUE type function can be applied only to those structure types that have at least one member with an INITIAL attribute.

IBM2211I S Shift-out code has no closing shift-in code before the right margin.

Explanation: Every DBCS shift-out code between the margins must have a matching DBCS shift-in code also between the margins.

IBM2212I S Argument to the *BUILTIN name* built-in must be a structure.

Explanation: The argument to the named built-in subroutine must be a structure.

IBM2213I S Block contains too many label arrays.

Explanation: Procedures and begin blocks must contain fewer than 2048 label arrays.

IBM2214I S *Attribute* is invalid on structure parents.

Explanation: The XMLATTR and XMLOMIT attributes may be used only on base structure elements.

IBM2215I S *Attribute* is invalid on unnamed structure elements.

Explanation: The XMLATTR and XMLOMIT attributes may be used only on named structure elements.

IBM2216I S *Attribute* is invalid on arrays.

Explanation: The XMLATTR and XMLOMIT attributes may be used only on scalar structure elements.

IBM2217I S XMLATTR is invalid if the previous element at that logical level does not also have the XMLATTR attribute.

Explanation: The XMLATTR attribute may be used on a structure element only if all its previous sister elements at the same logical level also had the XMLATTR attribute.

IBM2218I S *Attribute* is invalid on non-native FLOAT elements.

Explanation: The XMLOMIT attribute may not be used on FLOAT elements using a data representation not supported by the hardware.

IBM2219I S Parameters declared as INONLY must not contain any elements declared with the ASSIGNABLE attribute.

Explanation: If a parameter is declared as INONLY, then the ASSIGNABLE attribute is invalid on it and all of the elements it contains.

IBM2220I S Parameters declared as OUTONLY must contain at least one element declared with the ASSIGNABLE attribute.

Explanation: If a parameter is declared as OUTONLY, then the NONASSIGNABLE attribute must not be specified on all of its elements.

IBM2221I S A non-constant array extent in a BASED variable is invalid if the array has more than one dimension.

Explanation: The use of a non-constant extent in BASED variable without using REFER is limited. In an array, its use requires that the array has only one dimension.

IBM2222I S A non-constant array extent in a BASED variable is invalid if the array has a non-constant lower bound.

Explanation: The use of non-constant extents in BASED variables without using REFER is limited. In an array, its use requires that the array has a constant lower bound.

IBM2223I S A non-constant array extent in a BASED structure is invalid if any other fields in the structure have non-constant extents.

Explanation: The use of non-constant extents in BASED variables without using REFER is limited. In an array that is part of a structure, its use requires that no other field in the structure have non-constant extents.

IBM2224I S A non-constant AREA, BIT, GRAPHIC, or WIDECHAR extent in a BASED variable is invalid if the variable is an array element or part of a structure.

Explanation: The use of non-constant extents in BASED variables without using REFER is limited. In an AREA, BIT, GRAPHIC or WIDECHAR extent, its use requires that the AREA or string is a scalar.

IBM2225I S A non-constant CHARACTER extent in a BASED variable is invalid if the string is ALIGNED and either VARYING or VARYING4.

Explanation: The use of non-constant extents in BASED variables without using REFER is limited. In a CHARACTER extent, its use requires that the string be either UNALIGNED, NONVARYING or VARYINGZ.

IBM2226I S A non-constant array extent in a BASED variable is invalid if there are any sibling fields after the array or any of the array's parents.

Explanation: The use of non-constant extents in

BASED variables without using REFER is limited. In an array, its use requires that the array and the array's parents have no sibling fields.

IBM2227I S **A non-constant CHARACTER extent in a BASED structure is invalid if the string is a member of an array of structures.**

Explanation: The use of non-constant extents in BASED variables without using REFER is limited. In a CHARACTER extent, its use requires that the string not be part of an array.

IBM2228I S **A non-constant CHARACTER extent in a BASED structure is invalid unless the string is the last field in the structure and not part of a union.**

Explanation: The use of non-constant extents in BASED variables without using REFER is limited. In a CHARACTER extent, its use requires that the string be the last element in the structure and not part of a union.

IBM2230I S **The argument to the** *BUILTIN name* **built-in must have the attributes REAL FIXED BIN and scale factor zero.**

Explanation: This applies, for example, to the POPCNT built-in function.

IBM2231I S **The** *BUILTIN name* **built-in is supported only with the native character set.**

Explanation: The XMLCHAR built-in function cannot be used with DFT(EBCDIC) on Windows or AIX nor with DFT(ASCII) on the host.

IBM2232I S **There must be only one target in a BY DIMACROSS assignment.**

Explanation: Multiple targets are not permitted in BY DIMACROSS assignments. For example, the following is invalid.

```
dcl 1 a, 2 a1 fixed bin, 2 a2 fixed bin;
dcl 1 b like a;
dcl 1 c(100) dimacross like a;

a,b = c, by dimacross(jx);
```

IBM2233I S **The target in a BY DIMACROSS assignment must be a structure reference.**

Explanation: The target in a BY DIMACROSS assignment must not be an array of structures or a scalar. For example, the following is invalid.

```
dcl 1 a(100), 2 a1 fixed bin, 2 a2 fixed bin;
dcl 1 b(100) dimacross, 2 b1 fixed bin, 2 b2 fixed bin;

a = b, by dimacross(1);
```

IBM2234I S **No arrays are permitted in the source in a BY DIMACROSS assignment.**

Explanation: The source in a BY DIMACROSS assignment must not include any array references.

IBM2235I S **In a BY DIMACROSS assignment, the immediate children of any structure not declared with DIMACROSS must not be arrays.**

Explanation: The immediate children of a structure used in a BY DIMACROSS assignment must be scalars or substructures, but not arrays unless the structure was declared with the DIMACROSS attribute. For example, the following is invalid.

```
dcl 1 a, 2 a1(100) fixed bin, 2 a2(100) fixed bin;
dcl 1 b(100) dimacross, 2 b1 fixed bin, 2 b2 fixed bin;

a = b, by dimacross(1);
```

IBM2236I S *BUILTIN name* **argument must have the DIMACROSS attribute.**

Explanation: The named built-in function is valid only when applied to a reference to a variable declared with the DIMACROSS attribute.

IBM2237I S **The third argument to the ALLCOMPARE built-in must be a CHAR(2) constant.**

Explanation: The third argument to the ALLCOMPARE built-in function must be a restricted expression with the attributes CHAR(2) NONVARYING.

IBM2238I S **The third argument to the ALLCOMPARE built-in must specify the name of a comparison operator.**

Explanation: When uppercased, the third argument to the ALLCOMPARE built-in function must be one of 'EQ', 'LT', 'LE', 'GE', 'GT', or 'NE'.

IBM2239I S **Invalid use of unspecified STRUCT type** *type name*.

Explanation: If a DEFINE STRUCT statement specifies no member names, then any attempt to dereference the type is invalid.

IBM2240I S **Arithmetic operations are not allowed on handles for unspecified structure definitions.**

Explanation: The size of an unspecified structure is unknown, and hence all arithmetic operations on handles for it are ill-defined.

IBM2241I S **The argument to the type function** *type function* **must be a specified structure type name.**

Explanation: The argument to the named type function must be the name of a structure type that was fully specified.

IBM2242I S **Subtraction of HANDLE from HANDLE is invalid unless both point to the same type.**

Explanation: If h1 is a handle for structure type t1 and h2 is a handle for structure type t2, the h1-h2 is invalid unless t1 and t2 are the same.

IBM2243I S **The attributes derived from the PROCEDURE statement for the ENTRY constant** *variable name* **do not match those in its explicit declaration. RETURNS attributes do not match.**

Explanation: A label on a PROCEDURE statement constitutes a declaration for an ENTRY constant with that name. That name also appears in a DECLARE statement, but the attributes in those two declarations do not match.

IBM2244I S **The attributes derived from the PROCEDURE statement for the ENTRY constant** *variable name* **do not match those in its explicit declaration. BYVALUE/BYADDR attributes in RETURNS do not match.**

Explanation: A label on a PROCEDURE statement constitutes a declaration for an ENTRY constant with that name. That name also appears in a DECLARE statement, but the attributes in those two declarations do not match.

IBM2245I S **The attributes derived from the PROCEDURE statement for the ENTRY constant** *variable name* **do not match those in its explicit declaration. LINKAGE values do not match.**

Explanation: A label on a PROCEDURE statement constitutes a declaration for an ENTRY constant with that name. That name also appears in a DECLARE statement, but the attributes in those two declarations do not match.

IBM2246I S **The attributes derived from the PROCEDURE statement for the ENTRY constant** *variable name* **do not match those in its explicit declaration. OPTIONS values do not match.**

Explanation: A label on a PROCEDURE statement constitutes a declaration for an ENTRY constant with that name. That name also appears in a DECLARE statement, but the attributes in those two declarations do not match.

IBM2247I S **The attributes derived from the PROCEDURE statement for the ENTRY constant** *variable name* **do not match those in its explicit declaration. Parameter counts do not match.**

Explanation: A label on a PROCEDURE statement constitutes a declaration for an ENTRY constant with that name. That name also appears in a DECLARE statement, but the attributes in those two declarations do not match.

IBM2248I S **The attributes derived from the PROCEDURE statement for the ENTRY constant** *variable name* **do not match those in its explicit declaration. BYVALUE/BYADDR attributes in parameter** *parameter-number* **do not match.**

Explanation: A label on a PROCEDURE statement constitutes a declaration for an ENTRY constant with that name. That name also appears in a DECLARE statement, but the attributes in those two declarations do not match.

IBM2249I S **The attributes derived from the PROCEDURE statement for the ENTRY constant** *variable name* **do not match those in its explicit declaration. Number of dimensions for parameter** *parameter-number* **do not match.**

Explanation: A label on a PROCEDURE statement constitutes a declaration for an ENTRY constant with that name. That name also appears in a DECLARE statement, but the attributes in those two declarations do not match.

IBM2250I S **The attributes derived from the PROCEDURE statement for the ENTRY constant** *variable name* **do not match those in its explicit declaration. Lower bounds for parameter** *parameter-number* **do not match.**

Explanation: A label on a PROCEDURE statement constitutes a declaration for an ENTRY constant with that name. That name also appears in a DECLARE

statement, but the attributes in those two declarations do not match.

IBM2251I S **The attributes derived from the PROCEDURE statement for the ENTRY constant** *variable name* **do not match those in its explicit declaration. Upper bounds for parameter** *parameter-number* **do not match.**

Explanation: A label on a PROCEDURE statement constitutes a declaration for an ENTRY constant with that name. That name also appears in a DECLARE statement, but the attributes in those two declarations do not match.

IBM2252I S **The attributes derived from the PROCEDURE statement for the ENTRY constant** *variable name* **do not match those in its explicit declaration. Alignment of parameter** *parameter-number* **does not match.**

Explanation: A label on a PROCEDURE statement constitutes a declaration for an ENTRY constant with that name. That name also appears in a DECLARE statement, but the attributes in those two declarations do not match.

IBM2253I S **The attributes derived from the PROCEDURE statement for the ENTRY constant** *variable name* **do not match those in its explicit declaration. Number and attributes of structure members in parameter** *parameter-number* **do not match.**

Explanation: A label on a PROCEDURE statement constitutes a declaration for an ENTRY constant with that name. That name also appears in a DECLARE statement, but the attributes in those two declarations do not match.

IBM2254I S **The attributes derived from the PROCEDURE statement for the ENTRY constant** *variable name* **do not match those in its explicit declaration. Attributes of parameter** *parameter-number* **do not match.**

Explanation: A label on a PROCEDURE statement constitutes a declaration for an ENTRY constant with that name. That name also appears in a DECLARE statement, but the attributes in those two declarations do not match.

IBM2255I S **The argument to the** *BUILTIN name* **built-in must be numeric, bit, or character.**

Explanation: This message applies to the UTF8 built-in function. GRAPHIC and non-computational arguments are not allowed.

IBM2256I S **The result of the** *BUILTIN name* **built-in function would have a length greater than the the maximum allowed for a CHARACTER string.**

Explanation: Conversion of CHAR or WCHAR to UTF-8 can produce a result string that is longer than the source string because some CHAR(1) and WCHAR(1) values can produce CHAR(2) or CHAR(3) strings when converted to UTF-8. If there are too many of these values in the source string then the target string would have a length greater than the the maximum allowed for a CHARACTER string.

IBM2257I S **The argument to the** *BUILTIN name* **built-in function must hold valid UTF-16.**

Explanation: This message applies to the UTF8 built-in function.

IBM2258I S **The argument to the** *BUILTIN name* **built-in must have type CHARACTER.**

Explanation: This message applies to the UTF8TOCHAR and UTF8TOWCHAR built-in functions.

IBM2259I S **The argument to the** *BUILTIN name* **built-in must have hold valid UTF-8.**

Explanation: This message applies to the UTF8TOCHAR and UTF8TOWCHAR built-in functions.

IBM2260I S **INITIAL expressions in DEFINE STRUCT must not depend on any address values.**

Explanation: These expressions must be simple restricted expressions. For example, ENTRY, FILE and LABEL constants must not be used in these INITIAL expressions

IBM2261I S **Overpunch and currency characters are not allowed in WIDEPIC specifications.**

Explanation: These characters are allowed in PICTURE specifications, but not in WIDEPIC.

IBM2262I S A and X characters are not allowed in WIDEPIC specifications.

Explanation: These characters are allowed in PICTURE specifications, but not in WIDEPIC.

IBM2263I S REFER objects must not be COMPLEX WIDEPIC.

Explanation: REFER objects should have the REAL attribute.

IBM2264I S The *attribute* attribute is invalid in a LOCATES descriptor.

Explanation: The LOCATES descriptor may not specify a structure, union or array. The following code example is invalid:

```
dcl b offset(a) locates( 1 union, 2 ptr, 2 ptr );
```

IBM2265I S Extents in LOCATES descriptors must be constants.

Explanation: In LOCATES descriptors, any string length and AREA size must be specified with a restricted expression that has computational type.

IBM2266I S The argument to *BUILTIN name* built-in must have the LOCATES attribute.

Explanation: This rule applies to the LOCVAL and similar built-in functions.

IBM2267I S The first argument to *BUILTIN name* built-in must have the LOCATES attribute.

Explanation: This rule applies to the LOCNEWSPACE and similar built-in functions.

IBM2268I S Argument to the LOCVAL pseudovariable must have the LOCATES attribute.

Explanation: The LOCVAL pseudovariable can be applied only to variables with the LOCATES attribute.

IBM2269I S LOCATES attribute is valid only with OFFSET.

Explanation: The LOCATES attribute cannot be used on any other types.

IBM2270I S Only one description is allowed in a LOCATES descriptor.

Explanation: A located type can specify only one value. The following declaration is not correct:

```
dcl b offset(a) locates( ptr, ptr );
```

IBM2271I S The first argument to *BUILTIN name* built-in must be a scalar reference.

Explanation: This rule applies to the LOCNEWSPACE and similar built-in functions.

IBM2272I S The second argument to *BUILTIN name* built-in must be a scalar reference.

Explanation: This rule applies to the LOCNEWVALUE and similar built-in functions.

IBM2273I S The OFFSET argument to *BUILTIN name* built-in must have an AREA qualification.

Explanation: This rule applies to the LOCVAL and similar built-in functions.

IBM2274I S The second argument to *BUILTIN name* built-in must have the LOCATES attribute.

Explanation: This rule applies to the LOCNEWVALUE and similar built-in functions.

IBM2275I S Third argument to *BUILTIN name* built-in must have type AREA.

Explanation: This rule applies to the LOCNEWVALUE built-in functions.

IBM2276I S The argument to *BUILTIN name* built-in must have the LOCATES attribute or contain subelements with the LOCATES attribute.

Explanation: This rule applies to the LOCSTG and similar built-in functions.

IBM2277I S %INCLUDE statements are not allowed under NOINCLUDE.

Explanation: Under the NOINCLUDE compiler option, %INCLUDE statements are valid only if the MACRO preprocessor is used.

IBM2278I S Source is not valid UTF-8.

Explanation: The source file contains lines that would be rejected by the UVALID built-in function.

IBM2279I S *option* **option contains invalid UTF-8.**

Explanation: The specified option contains values that would be rejected by the UVALID built-in function.

IBM2280I S The corresponding characters in the two NAMES strings must have the same length.

Explanation: In the NAMES('abc', 'xyz') option, each of the UTF-8 characters in the first string must have a corresponding character of the same UTF-8 length in the second string.

IBM2281I S The first argument to *BUILTIN name* **built-in must have computational type or ordinal type.**

Explanation: An expression contains the named built-in function with the specified argument having a noncomputational type that is either not an ordinal type. This message applies to the INLIST and BETWEEN built-in functions.

IBM2282I S REINIT reference must be a level 1 item.

Explanation: In the statement REINIT x, x must not be a structure or union member.

IBM2283I S REINIT references must be BASED, AUTO, CTL or STATIC.

Explanation: In the statement REINIT x, x must not be DEFINED, constant, or a parameter.

IBM2284I S The first and second arguments to the *BUILTIN name* **built-in must have matching types.**

Explanation: This message applies to the LOCNEWVALUE built-in functions. In LOCNEWVALUE(x, y), if y has the attribute LOCATES(t) where t is an ORDINAL or STRUCT type, then x must have the same type.

IBM2285I S The argument to the *BUILTIN name* **built-in must have the attributes UNSIGNED REAL FIXED BIN(64,0).**

Explanation: This applies, for example, to the PLISTCK and PLISTCKF built-in subroutines.

IBM2286I S The argument to the *BUILTIN name* **built-in must have the attributes CHAR NONVARYING and length** *length***.**

Explanation: This applies, for example, to the PLISTCKE built-in subroutine where the argument must have length 16.

IBM2287I S Argument number *argument number* **to the** *BUILTIN name* **built-in must contain only standard computational types.**

Explanation: The JsonGetValue and similar built-in functions cannot be applied to aggregates or scalars containing noncomputational types or containing any COMPLEX numeric or any FIXED numeric with a scale factor that is either negative or larger than its preicison.

IBM2288I S Argument number *argument number* **to the** *BUILTIN name* **built-in must not be a UNION or contain any UNIONs.**

Explanation: The JsonGetValue and similar built-in functions cannot be applied to unions or to structures containing unions.

IBM2289I S Argument number *argument number* **to the** *BUILTIN name* **built-in must not contain any GRAPHIC elements.**

Explanation: The JsonGetValue and similar built-in functions cannot be applied to aggregates or scalars containing GRAPHIC data.

IBM2290I S Member *member-number* **in the event structure argument to the** *BUILTIN name* **built-in must have a BYADDR NATIVE FIXED BIN(63) as parameter number** *parameter-number***.**

Explanation: The indicated element of the structure supplying the event handlers for the SAX parser must have a BYADDR NATIVE FIXED BIN(63) in the specified parameter position. See the Programming Guide for more details.

IBM2291I S POINTER precision is invalid.

Explanation: In 64-bit mode, the only valid values for the POINTER precision are 32 and 64. Otherwise the only valid value is 32.

IBM2292I S Target in *statement* **statement must not be the name of a PROC or ENTRY statement.**

Explanation: The target in a FETCH or RELEASE statement must be outside the current compilation unit.

IBM2293I S **The *BUILTIN name* built-in is not supported under CMPAT(V1).**

Explanation: CMPAT(V2), CMPAT(V3) or CMPAT(LE) must be used when compiling any code using this built-in function.

IBM2294I S **A value greater than 32K for the STRING subooption of the LIMITS option is valid only under CMPAT(V3) and CMPAT(LE).**

Explanation: Strings longer than 32767 are not supported under CMPAT(V1) or CMPAT(V2).

IBM2295I S **A value greater than 32K for the STRING subooption of the LIMITS option is valid only under BIFPREC(31).**

Explanation: Strings longer than 32767 are not supported under BIFPREC(15).

IBM2296I S **Argument number *argument number* to *BUILTIN name* built-in must have the same ordinal type as the first argument.**

Explanation: An expression contains the named built-in function with the specified argument having either a non-ordinal type or an ordinal type that is not the same ordinal type as the first argument. This message applies to the INLIST and BETWEEN built-in functions.

IBM2297I S **The *BUILTIN name* built-in function is supported only under LP(64).**

Explanation: Built-in functions such as ALLOC31 are supported only under z/OS and only under the LP(64) option.

IBM2298I S **The *BUILTIN name* built-in function is supported only when the compiler option CHECK(STORAGE) is used.**

Explanation: Built-in functions such as ALLOCSIZE are supported only under the CHECK(STORAGE) compiler option.

IBM2299I S **No value can fall in the interval defined by the second and third arguments to the *BUILTIN name* built-in function.**

Explanation: The values a and b in BETWEEN(x,a,b) must satisfy a <= b. The values a and b in BETWEENEXCLUSIVE(x,a,b) must satisfy a < b, and the same is true for BETWEENLEFTEXCLUSIVE and BETWEENRIGHTEXCLUSIVE.

IBM2300I S **The compiler was disabled in the IFAPRDxx parmlib member. The compilation will terminate without further processing.**

Explanation: The SMF registration of the compiler failed because it has been disabled in the IFAPRDxx parmlib member.

IBM2301I S **The IFAEDREG registration of the compiler failed with return code *return code* . The compilation will terminate without further processing.**

Explanation: The SMF registration of the compiler failed with the indicated return code.

IBM2302I S **The option *option* is not supported under LP(64).**

Explanation: The specified option is not supported under LP(64). This is true, for example, of the SYSTEM(IMS) option.

IBM2303I S ***codepage* is not a supported codepage.**

Explanation: The specified value is not a supported codepage. See the Programming Guide for a list of the supported codepages.

IBM2304I S **The *attribute* attribute is not supported under CMPAT(V1).**

Explanation: CMPAT(V2), CMPAT(V3) or CMPAT(LE) must be used when compiling any code using this attribute.

IBM2305I S **The ASSERT COMPARE operator must be a CHAR(2) constant.**

Explanation: If an operator is specified in an ASSERT COMPARE statement, it must be a restricted expression with the attributes CHAR(2) NONVARYING.

IBM2306I S **The ASSERT COMPARE operator must specify the name of a comparison operator.**

Explanation: If an operator is specified in an ASSERT COMPARE statement, it must be one of 'EQ', 'LT', 'LE', 'GE', 'GT', or 'NE'.

IBM2307I S **The first argument to the *BUILTIN name* built-in must be a suitable one-dimensional array.**

Explanation: The array argument to the named built-in function must have exactly one dimension. For BINSEARCH and QUICKSORT, the array must consist of scalars. This message applies to the BINSEARCH,

BINSEARCHX, QUICKSORT, and QUICKSORTX built-in functions.

IBM2308I S **The first argument to the** *BUILTIN name* **built-in must be ALIGNED if NONVARYING BIT.**

Explanation: If the first argument to the named built-in function is NONVARYING BIT, then it must be ALIGNED. This message applies to the BINSEARCH, BINSEARCHX, QUICKSORT, and QUICKSORTX built-in functions.

IBM2309I S **Comparison in** *BUILTIN name* **built-in function is unsupported.**

Explanation: This message applies to the BINSEARCH built-in function and similar functions. The array and the search argument must be both string or REAL numeric, both ordinals of the same ordinal type, both pointers, or both handles to the same structure type.

IBM2310I S **The compare function passed to the** *BUILTIN name* **built-in must be a LIMITED ENTRY, must return BYVALUE a NATIVE FIXED BIN(31), must have exactly two BYVALUE POINTER arguments, and must have the OPTLINK linkage.**

Explanation: This message applies to the third argument to the BINSEARCHX built-in function and similar functions. This argument is the compare function to be invoked to compare elements during the binary search. It must be a LIMITED ENTRY (and hence must not be a nested PROCEDURE) and must have the other properties listed in the message.

IBM2311I S **Labels are not allowed on the END statement for a PACKAGE.**

Explanation: Labels must not be applied to the END statement for a PACKAGE.

IBM2312I S **Argument number** *argument number* **to** *BUILTIN name* **built-in must be a scalar expression.**

Explanation: An expression contains the named built-in function when the specified argument is an aggregate expression.

IBM2313I S **Argument number** *argument number* **to** *BUILTIN name* **built-in must be an array expression.**

Explanation: An expression contains the named built-in function when the specified argument is a scalar or structure expression.

IBM2314I S *BUILTIN name* **built-in does not support arrays of this type.**

Explanation: The QUICKSORT built-in supports only a limited set of array types. For example, FIXED BIN and ORDINAL arrays must be REAL and NATIVE.

IBM2315I S **Argument number** *argument number* **to** *BUILTIN name* **built-in must be REAL FIXED BIN with scale factor zero.**

Explanation: This message applies to the REGEX and other built-in functions where some arguments must have the attributes REAL FIXED BIN PRECISION(p,0).

IBM2316I S **Argument number** *argument number* **to** *BUILTIN name* **built-in must have CHARACTER type.**

Explanation: This message applies to the REGEX and other built-in functions where some arguments must have the CHARACTER attribute.

IBM2317I S **Argument number** *argument number* **to** *BUILTIN name* **built-in must be an ASSIGNABLE reference.**

Explanation: The indicated argument to the named built-in function must be an ASSIGNABLE reference so that it can be assigned a value. This message applies, for example, to the first two arguments of the REGEX built-in function.

IBM2318I S *attribute* **attribute is valid only with computational types.**

Explanation: The VALUELIST and VALUERANGE attributes cannot be used on non-computational tyeps.

IBM2319I S *attribute* **attribute is not valid with COMPLEX types.**

Explanation: The VALUERANGE attribute cannot be used on COMPLEX numeric tyeps.

IBM2320I S **First argument to** *BUILTIN name* **built-in must be a reference to a variable with the VALIDLIST or VALIDRANGE attribute.**

Explanation: The argument to the VALIDVALUE built-in function must have one of the above attributes so that its value can be checked against the declared list or range of values.

IBM2321I S *attribute* **contains duplicate values.**

Explanation: The items in VALUELIST and VALUERANGE lists should be unique.

IBM2322I S The second value in the VALUERANGE attribute must be larger than the first.

Explanation: The items in the VALUERANGE attribute should be in strictly ascending order. Both of the following are invalid.

```
dcl a fixed bin valuerange(12,1);
dcl b fixed bin valuerange(1,1);
```

IBM2323I S The second argument to *BUILTIN name* **built-in must have a type that is comparable to the first argument.**

Explanation: The arguments to the VALIDVALUE built-in function must be comparable. This means that if the first argument has a computational type, then the second must also, and if the first argument has an ordinal type, then the second must have the same ordinal type.

IBM2324I S The attributes derived from the PROCEDURE statement for the ENTRY constant *variable name* **do not match those in its explicit declaration. The EXTERNAL names do not match: one name is** *external name***, and the other is** *external name***.**

Explanation: A label on a PROCEDURE statement constitutes a declaration for an ENTRY constant with that name. That name also appears in a DECLARE statement, but the attributes in those two declarations do not match.

IBM2325I S The values specified for the ROUTCDE and DESC in a WTO must be between 1 and 16.

Explanation: These values specify which bits are set in the ROUTCDE and DESC fields when a WTO or WTOR is issued. These fields consist of 16 bits, and hence the values must be between 1 and 16.

Chapter 5. MACRO, CICS, and SQL Preprocessor Messages (3000-3999)

IBM3000I I *note*

Explanation: This message is used to report Db2 or CICS backend messages with a return code of 0.

IBM3019I I Program contains no EXEC SQL statements requiring translation.

Explanation: The SQL suboption has been specified for the PP option, but the program contains no EXEC SQL statements other than possibly EXEC SQL INCLUDE statements. The DBRMLIB will not be updated.

IBM3020I I Comment spans *line-count* **lines.**

Explanation: A comment ends on a different line than it begins. This may indicate that an end-of-comment delimiter is missing.

IBM3021I I String spans *line-count* **lines.**

Explanation: A string ends on a different line than it begins. This may indicate that a closing quote is missing.

IBM3024I I *note*

Explanation: This message is used by %NOTE statements with a return code of 0.

IBM3250I W *note*

Explanation: This message is used to report DB2 or CICS backend messages with a return code of 4.

IBM3251I W *identifier* **is multiply defined, but with different attributes. The declaration is ignored.**

Explanation: Attributes and declares must be consistent.

```
%a: proc;
%end;
%dcl a;
```

IBM3252I W The attribute *attribute* **conflicts with previous attributes and is ignored.**

Explanation: Attributes must be consistent.

```
dcl a fixed char;
```

IBM3253I W Comment spans more than one file.

Explanation: A comment ends in a different file than it begins. This may indicate that an end-of-comment statement is missing.

IBM3254I W String spans more than one file.

Explanation: A string ends in a different file than it begins. This may indicate that a closing quote is missing.

IBM3255I W Delimiter missing between *nondelimiter* **and** *nondelimiter***. A blank is assumed.**

Explanation: A delimiter (for example, a blank or a comma) is required between all identifiers and constants.

```
dcl 1 a, 2 b, 3c;
```

IBM3256I W Multiple closure of groups. END statements will be inserted to close intervening groups.

Explanation: Using one END statement to close more than one group of statements is permitted, but it may indicate a coding error.

IBM3257I W Missing *character* **assumed.**

Explanation: The indicated character is missing, and there are no more characters in the source. The missing character has been inserted by the parser in order to correct your source.

IBM3258I W Missing *character* **assumed before** *character***.**

Explanation: The indicated character is missing and has been inserted by the parser in order to correct your source.

```
%dcl jump fixed;
%skip
%jump = 2;
```

IBM3259I W *note*

© Copyright IBM Corp. 1999, 2018

Explanation: This message is used by %NOTE statements with a return code of 4.

IBM3260I W Syntax of the %CONTROL statement is incorrect.

Explanation: The %CONTROL statement must be followed by FORMAT or NOFORMAT option enclosed in parentheses and then a semicolon.

IBM3261I W The suboption *suboption* is not valid for the suboption *option* of the *option* option.

Explanation: A suboption of a suboption of an option is incorrect. The suboption may be unknown or outside the allowable range.

```
*process deprecate(stmt(test));
```

IBM3262I W The suboption *option* of the *option* option must be followed by a (possibly empty) parenthesized list.

Explanation: A suboption of an option has been incorrectly specified. It must be followed by a left parenthesis and then a (possibly empty) list of items and a closing right parenthesis.

```
*process deprecate(stmt);
```

IBM3265I W Number of lines specified with %SKIP must be between 0 and 999 inclusive.

Explanation: Skip amounts greater than 999 are not supported.

```
%skip(2000);
```

IBM3270I W 'EXEC CICS' encountered, but the CICS option is not in effect. Command ignored.

Explanation: The CICS option must be in effect if the source contains EXEC CICS statements.

IBM3271I W 'EXEC CSPM' encountered, but the CSPM option is not in effect. Command ignored.

Explanation: The CSPM option must be in effect if the source contains EXEC CSPM statements.

IBM3272I W 'EXEC DLI' encountered, but the DLI option is not in effect. Command ignored.

Explanation: The DLI option must be in effect if the source contains EXEC DLI statements.

IBM3281I W SELECT statement contains no WHEN or OTHERWISE clauses.

Explanation: WHEN or OTHERWISE clauses are not required on SELECT statements, but their absence may indicate a coding error.

IBM3283I W SELECT statement contains no WHEN clauses.

Explanation: SELECT statements do not require WHEN clauses, but their absence may indicate a coding error.

IBM3285I W FIXED BINARY constant contains too many digits. Excess nonsignificant digits will be ignored.

Explanation: A FIXED BINARY constant must contain 31 or fewer digits.

IBM3286I W FIXED DECIMAL constant contains too many digits. Excess nonsignificant digits will be ignored.

Explanation: The maximum precision for FIXED DECIMAL constants is specified by the FIXEDDEC suboption of the LIMITS compiler option.

IBM3287I W Mantissa in FLOAT BINARY constant contains more digits than the implementation maximum. Excess nonsignificant digits will be ignored.

Explanation: Float binary constants are limited to 64 digits.

IBM3288I W Mantissa in FLOAT DECIMAL constant contains more digits than the implementation maximum. Excess nonsignificant digits will be ignored.

Explanation: Float decimal constants are limited to 18 digits.

IBM3289I W FLOAT literal is too big for its implicit precision. An appropriate HUGE value is assumed.

Explanation: The precision for a float literal is implied by the number of digits in its mantissa. For instance 1e99 is implicitly FLOAT DECIMAL(1), but the value 1e99 is larger than the largest value a FLOAT DECIMAL(1) can hold.

IBM3291I W The OPTIONS option *option-name* conflicts with the LANGLVL compiler option. The option will be applied.

Explanation: The named option is not part of the PL/I

language definition as specified in the LANGLVL compiler option.

IBM3292I W *suboption* **is not a valid suboption for** *option***.**

Explanation: The specified suboption is not one of the supported suboptions of the named option.

```
*process pp(macro('fixed(long)'));
```

IBM3293I W A required suboption is missing for the *suboption* **option.**

Explanation: The named option requires a suboption.

```
*process pp(macro('fixed'));
```

IBM3294I W A closing parenthesis is missing in the specification of the *option* **option. One is assumed.**

Explanation: A closing parenthesis is missing in the specification of the named option.

```
*process pp(macro('fixed(bin'));
```

IBM3295I W *option* **is not a supported option.**

Explanation: The named option is not, in fact, an option.

```
*process pp(macro('float'));
```

IBM3299I W Syntax of the %LINE directive is incorrect.

Explanation: The %LINE directive must be followed, with optional intervening blanks, by a parenthesis, a line number, a comma, a file name and a closing parenthesis.

```
%line( 19, test.pli );
```

IBM3300I W *identifier* **has not been declared. CHARACTER attribute assumed.**

Explanation: All variables should be declared.

IBM3309I W Comparison of *BUILTIN name* **to a value it could not return is odd.**

Explanation: This message points to a likely programming error. For example, comparing SYSPOINTERSIZE to the value 32 is almost certainly an error since the only values SYSPOINTERSIZE could return are 4 and 8.

IBM3310I W First argument to *BUILTIN name* **built-in should have string type.**

Explanation: To eliminate this message, apply the CHAR or BIT built-in function to the first argument.

```
dcl i fixed bin;
display( substr(i,4) );
```

IBM3311I W Argument *number* **to the** *BUILTIN name* **built-in function is missing. A null value will be passed for the missing argument.**

Explanation: An argument to the function reference is missing. A null string or zero will be passed, as appropriate, for the missing argument.

```
%dcl a fixed;

%a = max(n,);
```

IBM3312I W LEAVE will exit noniterative DO-group.

Explanation: This message is not produced if the LEAVE statement specifies a label. In the following loop, the LEAVE statement will cause only the immediately enclosing DO-group to be exited; the loop will not be exited.

```
do i = 1 to n;
  if a(i) > 0 then
    do;
      call f;
      leave;
    end;
  else;
end;
```

IBM3313I W Result of comparison is always constant.

Explanation: This message is produced when a variable is compared to a constant equal to the largest or smallest value that the variable could assume. In the following loop, the variable x can never be greater than 99, and hence the implied comparison executed each time through the loop will always result in a '1'b.

```
do x pic'99';

do x = 1 to 99;
end;
```

IBM3314I W The reference *reference* could refer to a parent or its child, but the child is assumed.

Explanation: For a structure named X with first child named X, a reference to X would by PL/I rules be resolved to the parent. But references to structures containing structures are invalid in SQL statements and so the reference is assumed to refer to the child. The reference should be changed from X to X.X.

IBM3315I W The reference *reference* is an array of structures. Arrays of structures are not valid in SQL statements, but because this structure consists of only one element, the reference is treated as if it were a reference to its lone child.

Explanation: If a dimensioned structure named A consists of just one child B, a reference to A is treated as a reference to A.B.

IBM3316I W The reference *reference* is a structure containing an array. Structures containing arrays are not valid in SQL statements, but because this structure consists of only one element, the reference is treated as if it were a reference to its lone child.

Explanation: If a structure named A consists of just one child B and B is an array, a reference to A is treated as a reference to A.B.

IBM3317I W *note*

Explanation: This message is used to report DB2 message DSNH030I.

IBM3320I W RETURNS attribute in ENTRY declare ignored.

Explanation: ENTRY declares should not specify a RETURNS attribute. In the example below, the "returns(char)" should be omitted.

```
%dcl a entry returns( char );
```

IBM3321I W RETURNS option assumed to enclose attribute in PROCEDURE statement.

Explanation: In a PROCEDURE statement, any RETURNS attribute should be enclosed in parentheses following the RETURNS keyword. In the example below, the "char" attribute should be specified as "returns(char)".

```
%a: proc char ;
  return( '1729' );
%end;
```

IBM3322I W Argument list for PROCEDURE *identifier* is missing. It will be invoked without any arguments.

Explanation: References in open code to PROCEDUREs that have parameters should always include at least an empty argument list. For example, the "display(a)" below should be "display(a())".

```
%a: proc( x ) char ;
  dcl x char;
  return( '1729' );
%end;
%act a;

display( a );
```

IBM3323I W Too few arguments for PROCEDURE *identifier*. Null values will be passed for the missing arguments.

Explanation: There are too few arguments for the specified procedure. Null strings or zeros will be passed, as appropriate, for the missing arguments.

```
%a: proc( x ) char ;
  dcl x char;
  return( '1729' );
%end;
%act a;

display( a() );
```

IBM3324I W Too many arguments for PROCEDURE *identifier*. Excess ignored.

Explanation: There are too many arguments for the specified procedure. The excess arguments will be ignored.

```
%a: proc( x ) char ;
  dcl x char;
  return( '1729' );
%end;
%act a;

display( a(1,2) );
```

IBM3325I W No data attributes specified in declare for *identifier*.

Explanation: Preprocessor variables should be declared with an attribute such as CHAR or FIXED. This message could indicate that there is an extraneous

comma in the declare statement as in this example.

```
%dcl a, char;
```

IBM3326I W The LIKE reference is neither a structure nor a union.

Explanation: The LIKE reference cannot be a scalar or an array of scalars.

```
dcl
  a fixed bin,
  1 b like a;
```

IBM3327I W The LIKE reference is ambiguous.

Explanation: The LIKE reference needs enough qualification to be unique.

```
dcl
  1 x like b,
  1 a,
    2 b,
      3 c,
      3 d,
    2 e,
      3 f,
      3 g,
  1 h,
    2 b,
      3 j,
      3 k;
```

IBM3328I W Neither the LIKE reference nor any of its substructures can be declared with the LIKE attribute.

Explanation: LIKE from LIKE is not supported.

```
dcl
  1 a,
    2 b1 like c,
    2 b2 like c,
  1 c,
    2 d fixed bin,
    2 e fixed bin;
dcl
  1 x like a;
```

IBM3329I W The LIKE reference must not be a member of a structure or union declared with the LIKE attribute.

Explanation: LIKE from LIKE is not supported.

```
dcl
  1 a,
    2 b1 like c,
    2 b2 like c,
  1 c,
    2 d fixed bin,
    2 e fixed bin;
dcl
  1 x like a.b1;
```

IBM3330I W The LIKE reference is unknown.

Explanation: The LIKE reference must be known in the block containing the LIKE attribute specification.

IBM3331I W The INCLUDE file *filename* will be deprecated.

Explanation: The named INCLUDE file was specified in the INCLUDE suboption of the DEPRECATENEXT option, and so any attempt to include it is flagged.

IBM3332I W The END statement has no matching BEGIN, DO, PACKAGE, PROC, or SELECT. This may indicate a problem with the syntax of a previous statement.

Explanation: An END statement has been found that matches no previous statement. This may indicate that a previous statement has a syntax error such as a missing closing semicolon.

IBM3333I W One or more END statements are missing. This may indicate a problem with the syntax of a previous statement.

Explanation: The source ended without closing END statements for all the open statement groups. This may indicate that a previous statement has a syntax error such as a missing closing semicolon.

IBM3334I W The ENTRY named *variable* will be deprecated.

Explanation: The named ENTRY was specified in the ENTRY suboption of the DEPRECATENEXT option, and so any use of it is flagged.

IBM3500I E *note*

Explanation: This message is used to report DB2 or CICS backend messages with a return code of 8.

IBM3501I E *note*

Explanation: This message is used by %NOTE statements with a return code of 8.

IBM3502I E An integer with a K suffix must have no more than 7 digits.

Explanation: An integer of the form dddK must have no more than 7 digits. The specified value is replaced by 1K.

IBM3503I E In an integer with a K suffix the digits must specify a value less than or equal to 2097152.

Explanation: The largest accepted value for an integer with a K suffix is 2097152K. The specified value is replaced by 2097151K.

IBM3504I E An integer with an M suffix must have no more than 4 digits.

Explanation: An integer of the form dddM must have no more than 4 digits. The specified value is replaced by 1M.

IBM3505I E In an integer with an M suffix the digits must specify a value less than or equal to 2048.

Explanation: The largest accepted value for an integer with an M suffix is 2048M. The specified value is replaced by 2047M.

IBM3506I E An integer with a G suffix must have only 1 digit.

Explanation: An integer of the form dddG must have no more than 1 digit. The specified value is replaced by 1G.

IBM3507I E In an integer with an G suffix the digits must specify a value less than or equal to 2.

Explanation: The largest accepted value for an integer with an G suffix is 2G. The specified value is replaced by 1G.

IBM3508I E Numeric precision of 0 replaced by 1.

Explanation: Numeric precisions must be positive.

IBM3509I E DECLARE statement has invalid syntax. No variables in it may be used in EXEC SQL statements.

Explanation: Fix the DECLARE statement so that it is syntactically correct.

IBM3510I E *keyword* **statement is not allowed where an executable statement is required. A null statement will be inserted before the** *keyword* **statement.**

Explanation: In certain contexts, for example after an IF-THEN clause, only executable statements are permitted. A DECLARE, DEFINE, DEFAULT or FORMAT statement has been found in one of these contexts. A null statement, (a statement consisting of only a semicolon) will be inserted before the offending statement.

IBM3511I E COUNTER value would exceed 99999. It will be reset to 0.

Explanation: The COUNTER built-in function should not be invoked more than 99999 times.

IBM3512I E Multiple closure of groups is not allowed under RULES(NOMULTICLOSE).

Explanation: Under RULES(NOMULTICLOSE), there should be no multiple closure of groups in your source program.

IBM3514I E Second argument to *BUILTIN name* **built-in is negative. It will be changed to 0.**

Explanation: The second argument to built-in functions such as COPY and REPEAT must be nonnegative.

```
x = copy( y, -1 );
```

IBM3515I E Scale factor is bigger than 127. It is replaced by 127.

Explanation: Scale factors must be between -128 and 127 inclusive.

IBM3516I E Scale factor is less than -128. It is replaced by -128.

Explanation: Scale factors must be between -128 and 127 inclusive.

IBM3517I E Sole bound specified for dimension *dimension number* **of array** *variable name* **is less than 1. An upper bound of 1 is assumed.**

Explanation: The default lower bound is 1, but the upper bound must be greater than the lower bound.

```
dcl x(-5) fixed bin;
```

IBM3518I E *identifier* **does not conform to the NAMEPREFIX option.**

Explanation: If the NAMEPREFIX option is specified, the names of all macro variables and procedures must start with the character specified in that option.

IBM3519I E Characters in B3 literals must be 0-7.

Explanation: In a B3 literal, each character must be either 0-7.

IBM3520I E Structure level of 0 replaced by 1.

Explanation: Structure level numbers must be positive.

IBM3521I E Structure level greater than 255 specified. It is replaced by 255.

Explanation: The maximum structure level supported is 255.

```
dcl
  1 a,
    256 b,
    2 c,
```

IBM3522I E A DECIMAL exponent is required.

Explanation: An E in a FLOAT constant must be followed by at least one decimal digit (optionally preceded by a sign).

IBM3523I E A second argument to the *BUILTIN name* built-in must be supplied for arrays with more than one dimension. A value of 1 is assumed.

Explanation: The LBOUND, HBOUND, and DIMENSION built-in functions require two arguments when applied to arrays having more than one dimension.

```
dcl a(5,10) fixed bin;
do i = 1 to lbound(a);
```

IBM3524I E Second argument to *BUILTIN name* built-in is not positive. A value of 1 is assumed.

Explanation: The DIMENSION, HBOUND and LBOUND built-in functions require that the second argument be positive.

IBM3525I E Second argument to *BUILTIN name* built-in is greater than the number of dimensions for the first argument. A value of *dimension count* is assumed.

Explanation: The second argument to the LBOUND, HBOUND, and DIMENSION built-in functions must be no greater than the number of dimensions of their array arguments.

```
dcl a(5,10) fixed bin;
do i = 1 to lbound(a,3);
```

IBM3526I E Repeated declaration of *identifier* is invalid and will be ignored.

Explanation: Level 1 variable names must not be repeated in the same block.

```
dcl a char, a fixed;
```

IBM3527I E Missing THEN assumed.

Explanation: THEN keyword must be part of any IF statement.

IBM3528I E Duplicate specification of arithmetic precision. Subsequent specification ignored.

Explanation: The precision attribute must be specified only once in a DECLARE statement.

```
dcl a fixed(15) bin(31);
```

IBM3529I E Scale factors are not allowed in FLOAT declarations.

Explanation: Scale factors are valid only in declarations of FIXED BIN or FIXED DEC. The first declaration below is invalid and should be changed to one of the subsequent declarations.

```
dcl a1 float dec(15,2);

dcl a2 fixed dec(15,2);
dcl a3 float dec(15);
```

IBM3530I E *identifier* is an array. ACTIVATE and DEACTIVATE are invalid for arrays.

Explanation: Only scalars may be activated.

IBM3531I E *identifier* is a statement label. ACTIVATE and DEACTIVATE are invalid for labels.

Explanation: Labels may not be activated.

IBM3533I E THEN clause outside of an open IF statement is ignored.

Explanation: THEN clauses are valid only immediately after an IF <expression>.

```
%if a > b; %then;
```

IBM3534I E ELSE clause outside of an open IF-THEN statement is ignored.

Explanation: ELSE clauses are valid only immediately after an IF-THEN statement.

```
do; if a > b then; end; else a = 0;
```

IBM3536I E END label is not a label on any open group.

Explanation: A Label on END statement must match a LABEL on an open DO, PROCEDURE, or SELECT statement.

```
a: do;
  ...
end b;
```

IBM3537I E An END statement may be missing after an OTHERWISE unit. One will be inserted.

Explanation: After an OTHERWISE unit in a SELECT statement, only an END statement is valid.

```
select;
  when ( ... )
    do;
    end;
  otherwise
    do;
    end;
display( .... );
```

IBM3538I E %END statement found without any open %PROCEDURE, %DO or %SELECT statements. It will be ignored.

Explanation: Any %END statement should be part of a %PROCEDURE-%END, %DO-%END or %SELECT-%END group.

IBM3539I E STRINGSIZE condition raised while evaluating expression. Result is truncated.

Explanation: During the conversion of a user expression during the compilation, the target string was found to be shorter than the source, thus causing the STRINGSIZE condition to be raised.

IBM3540I E STRINGRANGE condition raised while evaluating expression. Arguments are adjusted to fit.

Explanation: If all the arguments in a SUBSTR reference are constants or restricted expressions, the reference will be evaluated at compile- time and the STRINGRANGE condition will occur if the arguments do not comply with the rules described for the SUBSTR built-in function.

```
a = substr( 'abcdef', 5, 4 );
```

IBM3542I E LEAVE/ITERATE label is not a label on any open DO group.

Explanation: LEAVE/ITERATE must specify a label on an open DO loop.

```
%a: do jx = 1 to 1729;
  %leave b;
%end;
```

IBM3543I E ITERATE/LEAVE statement is invalid outside an open DO statement. The statement will be ignored.

Explanation: ITERATE/LEAVE statements are valid only inside DO groups.

```
%a: do jx = 1 to 1729;
%end;
%leave a;
```

IBM3544I E GX literals should contain a multiple of 4 hex digits.

Explanation: GX literals must represent graphic strings and hence must contain a multiple of 4 hex digits.

```
x = '00'gx;
```

IBM3545I E Upper bound for dimension *dimension number* **of array** *variable name* **is less than lower bound. Bounds will be reversed.**

Explanation: A variable has been declared with an upper bound that is less than its lower bound. The upper and lower bounds will be swapped in order to correct this. For example, DECLARE x(3:1) will be changed to DECLARE x(1:3).

IBM3546I E Identifier is too long. It will be collapsed to *identifier***.**

Explanation: All identifiers must be contained in 31 bytes or less. PL/I DBCS identifiers must have 14 or fewer DBCS characters.

IBM3547I E B assumed to complete iSUB.

Explanation: There is no language element of the form 1su.

```
dcl a(10) def b(1su, 1sub );
```

IBM3548I E Digit in BINARY constant is not zero or one.

Explanation: In a BINARY constant, each digit must be a zero or one.

IBM3549I E Characters in BIT literals must be 0 or 1.

Explanation: In a BIT literal, each character must be either zero or one.

IBM3550I E Character with decimal value *n* does not belong to the PL/I character set. It will be ignored.

Explanation: The indicated character is not part of the PL/I character set. This can occur if a program containing NOT or OR symbols is ported from another machine and those symbols are translated to a character that is not part of the PL/I character set. Using the NOT and OR compiler options can help avoid this problem.

IBM3551I E Characters in hex literals must be 0-9 or A-F.

Explanation: In a hex literal, each character must be either 0-9 or A-F.

IBM3552I E The statement element *character* is invalid. The statement will be ignored.

Explanation: The statement entered could not be parsed because the specified element is invalid.

IBM3553I E Use of underscore as initial character in an identifier accepted although invalid under LANGLVL(SAA).

Explanation: Under LANGLVL(SAA), identifiers must start with an alphabetic character or with one of the extralingual characters. They may not start with an underscore. Under LANGLVL(SAA2), identifiers may start with an underscore, although names starting with _IBM are reserved for use by IBM.

IBM3556I E Character with decimal value *n* does not belong to the PL/I character set. It is assumed to be an OR symbol.

Explanation: The indicated character is not part of the PL/I character set, but was immediately followed by the same character. This can occur if a program containing an OR symbol is ported from another machine and this symbol is translated to a character that is not part of the PL/I character set. Using the OR compiler option can help avoid this problem.

IBM3557I E Character with decimal value *n* does not belong to the PL/I character set. It is assumed to be a NOT symbol.

Explanation: The indicated character is not part of the PL/I character set, but was immediately followed by an =, < or > symbol. This can occur if a program containing a NOT symbol is ported from another machine and this symbol is translated to a character that is not part of the PL/I character set. Using the NOT compiler option can help avoid this problem.

IBM3558I E WX literals should contain a multiple of 4 hex digits.

Explanation: WX literals must represent unicode strings and hence must contain a multiple of 4 hex digits.

```
x = '00'wx;
```

IBM3565I E Statement type resolution requires too many lexical units to be examined. The statement will be ignored.

Explanation: To determine if a statement is an assignment or another PL/I statement, many elements of the statement may need to be examined. If too many have to be examined, the compiler will flag the statement as in error. For instance, the following statement could be a DECLARE until the equal sign is encountered by the lexer.

```
dcl ( a, b, c ) = d;
```

IBM3567I E Statements inside a SELECT must be preceded by a WHEN or an OTHERWISE clause.

Explanation: A WHEN or OTHERWISE might be missing.

```
select;
  i = i + 1;
  when ( a > 0 )
    ...
```

IBM3568I E Under RULES(NOLAXFIELDS), EXEC SQL SELECT statements must specify a list of field names.

Explanation: Under RULES(NOLAXFIELDS), EXEC SQL SELECT must be followed by one or more field names, not by an asterisk.

IBM3569I E Under RULES(NOLAXFIELDS), EXEC SQL INSERT INTO statements must specify a list of field names.

Explanation: Under RULES(NOLAXFIELDS), EXEC SQL INSERT INTO <table-name> must be followed by one or more field names.

IBM3570I E Extent expression is negative. It will be replaced by the constant 1.

Explanation: Extents must be positive.

```
dcl x char(-10);
```

IBM3571I E The SQL and PL/I float options are inconsistent.

Explanation: The compiler option DEFAULT(IEEE|HEXADEC) does not match the SQL preprocessor option FLOAT(IEEE|S390). Make sure they are consistent and resubmit your job.

IBM3572I E Initial level number in a structure is not 1.

Explanation: The level-1 DECLARE statement might be missing.

```
dcl
  2 a,
    3 b,
    3 c,
```

IBM3573I E Elements with level numbers greater than 1 follow an element without a level number. A level number of 1 is assumed.

Explanation: A structure level is probably missing.

```
dcl
    a,
    2 b,
    2 c,
```

IBM3574I E Variables declared without a name must be structure members or followed by a substructure list.

Explanation: The use of an asterisk in place of a name is permitted only for structure or union names or for members of structures or unions. An asterisk must not be used for a level-1 structure name that specifies the LIKE attribute.

```
dcl a fixed bin(15), * char(20) static init('who can use me');
```

IBM3575I E Duplicate specification of *attribute*. Subsequent specification ignored.

Explanation: Attributes such as CHAR must not be repeated for an element of a DECLARE statement.

```
dcl a char(10) char(20);
```

IBM3576I E The SQL statement is empty and is ignored.

Explanation: EXEC SQL statements must consist of more than merely EXEC SQL.

IBM3577I E INCONLY option is ignored because preceded by other options.

Explanation: The INCONLY option must be specified without any other options.

IBM3580I E Parameter *keyword* may not be set more than once. First setting is assumed.

Explanation: In a statement-form procedure invocation, each parameter must be specified only once. Any subsequent specifications will be ignored. In the example code, 17 would be returned for both invocations of P.

```
%p: proc( a ) stmt returns( char );
   dcl a char;
   return( a );
%end;
%act p;

display( p a(17) a(29); );

display( p(17) a(29); );
```

IBM3581I E Unknown keyword in statement-form procedure invocation. *keyword* and any argument are ignored.

Explanation: In a statement-form procedure invocation, any keyword specified must be the name of a parameter for that procedure.

```
%p: proc( a ) stmt returns( char );
   dcl a char;
   return( a );
%end;
%act p;

display( p a(17) b(29); );
```

IBM3582I E Parameter *identifier* is not declared.

Explanation: Each parameter in a procedure should be declared.

```
%a: proc( b, c );
  dcl b fixed;
%end;
```

IBM3583I E Labels on *keyword* statements are invalid and ignored.

Explanation: Labels are not permitted on DECLARE statements or on WHEN and OTHERWISE clauses.

IBM3589I E The identifier *identifier* is not the name of a built-in function. The BUILTIN attribute will be ignored.

Explanation: The BUILTIN attribute can be applied only to identifiers that are the names of built-in functions or subroutines.

IBM3590I E The attribute *keyword* is not supported and will be ignored.

Explanation: The named attribute is not supported by the macro facility.

```
%dcl a char external;
```

IBM3591I E Right parenthesis will be assumed at end of argument list.

Explanation: A right parenthesis is probably missing. If this occurs in the source, all the characters after the unmatched left parenthesis in the source will be interpreted as parameters to the function. If this occurs in a replacement string, all the characters after the unmatched left parenthesis in the string will be interpreted as parameters to the function.

IBM3603I E The end of the source was reached before the logical end of the program. Null statements and END statements will be inserted as necessary to complete the program.

Explanation: The source should contain END statements for all PROCEDUREs, DO groups, and SELECT statements, as well as statements for all IF-THEN and ELSE clauses.

IBM3604I E The procedure name *proc-name* has already been declared. The explicit declaration of the procedure name will not be accepted.

Explanation: Declarations for internal procedures are not permitted.

```
a: proc;
  dcl b entry options(byvalue);
  b: proc;
```

IBM3605I E The *type type* type *type name* is already defined. The redefinition is ignored.

Explanation: An ORDINAL type may be defined only once in any block.

IBM3606I E Repeated declaration of *identifier* is invalid. The name will be replaced by an asterisk.

Explanation: The variable names at any given sublevel within a structure or union must be unique.

```
dcl 1 a, 2 b fixed, 2 b float;
```

IBM3607I E UNSIGNED attribute for *type type* type *type name* conflicts with negative INITIAL values and is ignored.

Explanation: If an ORDINAL type is declared with the UNSIGNED attribute, any INITIAL values specified must be nonnegative.

IBM3608I E PRECISION specified for *type type* type *type name* is too small to cover its INITIAL values and is adjusted to fit.

Explanation: An ORDINAL type must have a precision larger enough to cover the range of values defined for it.

```
define ordinal
  colors
    ( red      init(0),
      orange   init(256)
      yellow   init(512) ) unsigned prec(8);
```

IBM3609I E A SELECT statement may be missing. A SELECT statement, without an expression, will be inserted.

Explanation: A WHEN or OTHERWISE clause has been found outside of a SELECT statement.

IBM3610I E Semicolon inserted after ELSE keyword.

Explanation: An END statement enclosing a statement such as DO or SELECT has been found before the statement required after ELSE.

```
do;
  if a > b then
    ...
  else
end;
```

IBM3612I E Semicolon inserted after OTHERWISE keyword.

Explanation: An END statement might be misplaced or a semicolon might be missing.

IBM3613I E Semicolon inserted after THEN keyword.

Explanation: An END statement might be misplaced or a semicolon might be missing.

IBM3614I E Semicolon inserted after WHEN clause.

Explanation: An END statement might be misplaced or a semicolon might be missing.

IBM3615I E Source file does not end with the logical end of the program.

Explanation: The source file contains statements after the END statement that closed the first PACKAGE or PROCEDURE. These statements will be ignored, but their presence may indicate a programming error.

IBM3616I E Subscripts have been specified for the variable *variable name*, but it is not an array variable.

Explanation: Subscripts can be specified only for elements of an array.

IBM3617I E Second argument in SUBSTR reference is less than 1. It will be replaced by 1.

Explanation: Otherwise the STRINGRANGE condition would be raised.

IBM3618I E Second argument in SUBSTR reference is too big. It will be trimmed to fit.

Explanation: Otherwise the STRINGRANGE condition would be raised.

IBM3619I E Third argument in SUBSTR reference is less than 0. It will be replaced by 0.

Explanation: Otherwise the STRINGRANGE condition would be raised.

IBM3620I E Third argument in SUBSTR reference is too big. It will be trimmed to fit.

Explanation: Otherwise the STRINGRANGE condition would be raised.

IBM3621I E More than 15 dimensions have been specified. Excess will be ignored.

Explanation: The maximum number of dimensions allowed for a variable, including all inherited dimensions, is 15.

IBM3624I E End-of-comment marker found when there are no open comments. Marker will be ignored.

Explanation: An */ was found when there was no open comment.

IBM3625I E There is no compiler directive *directive*. Input up to the next semicolon will be ignored.

Explanation: See the Language Reference Manual for the list of supported compiler directives.

IBM3626I E Listing control statement must start with a percent symbol.

Explanation: A listing control statement, even when in a preprocessor procedure, must be preceded by a "%".

```
%a: proc;
  skip;
%end;
```

IBM3628I E X literals should contain a multiple of 2 hex digits.

Explanation: An X literal may not contain an odd number of digits.

IBM3638I E Excess arguments for ENTRY *ENTRY name* ignored.

Explanation: More arguments were specified in an ENTRY reference than were defined as parameters in that ENTRY's declaration.

```
dcl e entry( fixed bin );
call e( 1, 2 );
```

IBM3639I E Excess arguments for *BUILTIN name* built-in ignored.

Explanation: More arguments were specified for the indicated built-in function than are supported by that built-in function.

```
i = acos( j, k );
```

IBM3640I E The attribute *attribute* is invalid if it is not followed by an element with a greater logical level.

Explanation: The named attribute is valid only on parent structures.

```
dcl
  1 a,
    2 b union,
      2 c1  fixed bin(31),
      2 c2  float bin(21),
    ...
```

IBM3641I E Level number following LIKE specification is greater than the level number for the LIKE specification. LIKE attribute is ignored.

Explanation: LIKE cannot be specified on a parent structure or union.

```
dcl
  1 a like x,
    2 b,
    2 c,
```

IBM3650I E *keyword* keyword accepted although invalid under LANGLVL(SAA).

Explanation: The indicated keyword (UNSIGNED in the example below) is not defined in the SAA level-1 language.

```
dcl x fixed bin unsigned;
```

IBM3651I E Use of S, D and Q constants accepted although invalid under LANGLVL(SAA).

Explanation: The definition of the SAA level-1 language does not include S, D, and Q floating-point constants.

IBM3652I E Use of underscores in constants accepted although invalid under LANGLVL(SAA).

Explanation: The definition of the SAA level-1 language does not permit using underscores in numeric and hex constants.

IBM3653I E Use of asterisks for names in declares accepted although invalid under LANGLVL(SAA).

Explanation: The definition of the SAA level-1 language does not permit using asterisks for structure element names.

IBM3654I E Use of XN constants accepted although invalid under LANGLVL(SAA).

Explanation: The definition of the SAA level-1 language does not include XN constants.

IBM3656I E Use of 3 arguments with *BUILTIN name* built-in accepted although invalid under LANGLVL(SAA).

Explanation: Under LANGLVL(SAA), the VERIFY and INDEX built-in functions are supposed to have exactly 2 arguments.

```
i = verify( s, j, k );
```

IBM3657I E Use of 1 argument with *BUILTIN name* built-in accepted although invalid under LANGLVL(SAA).

Explanation: Under LANGLVL(SAA), the DIM, LBOUND and HBOUND built-in functions are supposed to have 2 arguments.

```
i = dim( a );
```

IBM3658I E The INCLUDE file *filename* has been deprecated.

Explanation: The named INCLUDE file was specified in the INCLUDE suboption of the DEPRECATE option, and so any attempt to include it is flagged.

IBM3659I E The EXEC SQL *statement* statement has been deprecated.

Explanation: The named statement was specified in the STMT suboption of the DEPRECATE option, and so any occurrence of it is flagged.

IBM3660I E **The ENTRY named *variable* has been deprecated.**

Explanation: The named ENTRY was specified in the ENTRY suboption of the DEPRECATE option, and so any use of it is flagged.

IBM3661I E **Invalid use of question mark.**

Explanation: Question marks are valid in the source only if part of one of the trigraphs ??(or ??).

IBM3750I S *note*

Explanation: This message is used to report DB2 or CICS backend messages with a return code of 12.

IBM3751I S **A colon in an EXEC SQL statement must be followed by an identifier that starts a host variable reference.**

Explanation: A colon in an EXEC SQL statement must be followed by a host variable reference, and such a reference must start with an identifier.

IBM3752I S **Dot-qualified reference implies too many structure levels.**

Explanation: Structures are limited to at most 15 logical levels, and so any dot-qualified reference must have at most 14 dots (or else it would imply the structure had at least 16 logical levels).

IBM3753I S **Length in SQL TYPE IS *type name* is too large.**

Explanation: The maximum length for BIN is 255 and for VARBINARY 32704. See the Programming Guide for the maximum lengths for BLOBs, CLOBs, and DBCLOBs.

IBM3754I S **SQL TYPE IS *type name* must be followed by an opening left parenthesis.**

Explanation: The correct syntax is SQL TYPE IS type(length).

IBM3755I S **SQL TYPE IS *type name* must have an integer specifying its length after the opening left parenthesis.**

Explanation: The correct syntax is SQL TYPE IS type(length).

IBM3756I S **SQL TYPE IS *type name* must have a closing right parenthesis after the integer specifying its length.**

Explanation: The correct syntax is SQL TYPE IS type(length).

IBM3757I S **SQL TYPE IS XML AS *type name* must be followed by an opening left parenthesis.**

Explanation: The correct syntax is SQL TYPE IS XML AS type(length).

IBM3758I S **SQL TYPE IS XML AS *type name* must have an integer specifying its length after the opening left parenthesis.**

Explanation: The correct syntax is SQL TYPE IS XML AS type(length).

IBM3759I S **SQL TYPE IS XML AS *type name* must have a closing right parenthesis after the integer specifying its length.**

Explanation: The correct syntax is SQL TYPE IS XML AS type(length).

IBM3760I S **Too few arguments have been specified for the ENTRY *ENTRY name*.**

Explanation: The number of arguments must match the number of parameters in the ENTRY declaration.

IBM3761I S **Procedures may not be nested.**

Explanation: Macro procedures may not be nested.

IBM3762I S **No percent statements are allowed inside procedures.**

Explanation: Inside a procedure, statements should not begin with a percent. The %DCL in the example below should be just DCL.

```
%a: proc( x ) returns( char );
  %dcl x char;
  return( '<' || x || '>' );
%end;
```

IBM3763I S **Not enough virtual memory is available to continue the compile.**

Explanation: The compilation requires more virtual memory than is available. It may help to specify one or more of the following compiler options: NOINSOURCE, NOXREF, NOATTRIBUTES, and/or NOAGGREGATE

IBM3764I S *BUILTIN name* **argument must be a parameter.**

Explanation: An expression contains the named built-in function with an argument that is not a parameter.

IBM3765I S *BUILTIN name* **argument must be a reference.**

Explanation: An expression contains the named built-in function with an argument that is not a reference.

IBM3766I S **Aggregate contains more than 15 logical levels.**

Explanation: The maximum physical level allowed is 255, but the maximum logical level is 15.

IBM3767I S **Length in SQL TYPE IS** *type name* **must be greater than zero.**

Explanation: The length in BIN, VARBIN, BLOB, CLOB, and DBCLOB types must be positive.

IBM3768I S **The use of asterisks as subscripts is not permitted in the macro facility.**

Explanation: In the macro facility, all subscripts must be scalar expressions.

IBM3769I S **Argument to** *BUILTIN name* **built-in must have type CHARACTER(1) NONVARYING.**

Explanation: This applies to the RANK built-in function.

IBM3770I S **First argument to** *BUILTIN name* **built-in must be an array.**

Explanation: An expression contains the named built-in function with a first argument that is not an array. This message applies, for instance, to the DIMENSION, HBOUND, and LBOUND built-in functions.

IBM3771I S *note*

Explanation: This message is used by %NOTE statements with a return code of 12.

IBM3772I S **Third argument to** *BUILTIN name* **built-in would force STRINGRANGE.**

Explanation: If a third argument is given for one of the built-in functions INDEX or VERIFY, it must be positive.

IBM3773I S **Second argument to** *BUILTIN name* **built-in must be nonnegative.**

Explanation: The second argument for the built-in functions CHARACTER, BIT, and GRAPHIC must be zero or greater.

IBM3774I S **Too few arguments have been specified for the** *BUILTIN name* **built-in.**

Explanation: Supply the minimum number of arguments required.

IBM3775I S **The** *preprocessor name* **preprocessor requires the DFT(EBCDIC) option.**

Explanation: The use of the DFT(ASCII) option with either the CICS or SQL preprocessor is not supported.

IBM3778I S **Syntax of the %INCLUDE statement is incorrect.**

Explanation: %INCLUDE must be followed by a name and either a semicolon or else a second name in parenthesis and then a semicolon.

IBM3779I S **File specification after %INCLUDE is too long.**

Explanation: The maximum length of the file specification is 8 characters.

IBM3780I S **File specification missing after %INCLUDE.**

Explanation: %INCLUDE must be followed by a file name, not just a semicolon.

IBM3781I S **Procedures may have no more than 63 parameters.**

Explanation: The excess parameters will be removed from the proc statement.

IBM3782I S **SQL TYPE IS XML must be followed by the keyword AS.**

Explanation: The correct syntax is SQL TYPE IS XML AS type(length).

IBM3783I S **SQL TYPE IS XML AS must be followed by a valid type name.**

Explanation: The correct syntax is SQL TYPE IS XML AS type(length).

IBM3784I S **SQL TYPE IS TABLE must be followed by the keyword LIKE.**

Explanation: The correct syntax is SQL TYPE IS TABLE LIKE table-name AS LOCATOR.

IBM3785I S **SQL TYPE IS TABLE LIKE must be followed by a table name.**

Explanation: The correct syntax is SQL TYPE IS TABLE LIKE table-name AS LOCATOR.

IBM3786I S **SQL TYPE IS TABLE LIKE must be followed by the keyword AS after the table name.**

Explanation: The correct syntax is SQL TYPE IS TABLE LIKE table-name AS LOCATOR.

IBM3787I S **SQL TYPE IS TABLE must be followed by the keyword LOCATOR after the table name and the AS keyword.**

Explanation: The correct syntax is SQL TYPE IS TABLE LIKE table-name AS LOCATOR.

IBM3788I S **SQL TYPE IS must be followed by a valid type name.**

Explanation: The keywords SQL TYPE IS must be followed by a type name such as XML.

IBM3789I S **Index number** *index number* **into the variable** *variable name* **is less than the lower bound for that dimension.**

Explanation: Executing such a statement would most likely cause a protection exception.

```
%dcl a(5:10)  fixed;

%a(1) = 0;
```

IBM3790I S **Index number** *index number* **into the variable** *variable name* **is greater than the upper bound for that dimension.**

Explanation: Executing such a statement would most likely cause a protection exception.

```
%dcl a(5:10)  fixed;

%a(20) = 0;
```

IBM3791I S **Each dimension of an array must contain no more than 2147483647 elements.**

Explanation: It must be possible to compute the value of the DIMENSION built-in function for an array. For example, in DECLARE A(x:y), (y-x+1) must be less than 214748648.

IBM3792I S **Array** *variable name* **has too many elements. Bounds set to 1.**

Explanation: Arrays are limited to 2**20 elements.

IBM3793I S **Too few subscripts specified for the variable** *variable name***.**

Explanation: The number of subscripts given for a variable must match that variable's number of dimensions

IBM3794I S **Too many subscripts specified for the variable** *variable name***.**

Explanation: The number of subscripts given for a variable must match that variable's number of dimensions

IBM3795I S **Shift-out code has no closing shift-in code before the right margin.**

Explanation: Every DBCS shift-out code between the margins must have a matching DBCS shift-in code also between the margins.

IBM3796I S **Array expressions cannot be assigned to non-arrays, and if any target in a multiple assignment is an array, then all the targets must be arrays.**

Explanation: Array expressions may not, for instance, be assigned to structures or scalars.

IBM3797I S **RETURN statement without an expression is invalid inside a PROCEDURE that specified the RETURNS attribute.**

Explanation: All RETURN statements inside functions must specify a value to be returned.

```
%a: proc returns( fixed );

  return;
%end;
```

IBM3798I S **RETURN statement with an expression is invalid inside a PROCEDURE that did not specify the RETURNS attribute.**

Explanation: A statement of the form RETURN(x) is valid inside only PROCEDUREs that are defined with a RETURNS attribute.

```
%a: proc;

  return( 'this is invalid' );
%end;
```

IBM3799I S The DECLARE statement for the host variable *reference* is not inside an SQL DECLARE SECTION.

Explanation: Under the SQL option STDSQL(YES), all host variables must be declared between SQL BEGIN DECLARE SECTION and SQL END DECLARE SECTION statements.

IBM3800I S Function *function name* contains no RETURN statement.

Explanation: Functions must contain at least one RETURN statement.

IBM3801I S Target in assignment is invalid.

Explanation: The target in an assignment must be character or fixed element reference. Pseudovariables are not supported.

IBM3802I S Statement labels may not be used in expressions.

Explanation: Statement labels must be used only in GOTO, LEAVE and ITERATE statements.

IBM3803I S Target in concatenate-equals assignment must have type char.

Explanation: Compound concatenate assignments with fixed targets are not supported.

```
%dcl a fixed;

%a = '0';
%a ||= '1';
```

IBM3804I S Target in arithmetic-equals assignment must have type fixed.

Explanation: Compound arithmetic assignments with character targets are not supported.

```
%dcl a char;

%a = '0';
%a += '1';
```

IBM3805I S SQL TYPE IS XML *type* must be followed by the keyword LARGE.

Explanation: The correct syntax is SQL TYPE IS XML AS type LARGE OBJECT(length).

IBM3806I S SQL TYPE IS XML *type* LARGE must be followed by the keyword OBJECT.

Explanation: The correct syntax is SQL TYPE IS XML AS type LARGE OBJECT(length).

IBM3807I S SQL TYPE IS CHARACTER must be followed by the keyword LARGE.

Explanation: The correct syntax is SQL TYPE IS CHARACTER LARGE OBJECT(length).

IBM3808I S SQL TYPE IS BINARY must be followed by the keyword LARGE or by a length enclosed in parentheses.

Explanation: The correct syntax is SQL TYPE IS BINARY LARGE OBJECT(length) or SQL TYPE IS BINARY(length).

IBM3809I S SQL TYPE IS *type* LARGE must be followed by the keyword OBJECT.

Explanation: The correct syntax is SQL TYPE IS type LARGE OBJECT(length).

IBM3810I S Statement has too many labels.

Explanation: The compiler's limit on the number of labels on a statement has been exceeded. Reduce the number of labels on the statement.

IBM3811I S Expression contains too many nested subexpressions.

Explanation: The compiler's space for evaluating expressions has been exhausted. Rewrite the expression in terms of simpler expressions.

IBM3812I S Result of concatenating a string of length *string length* to a string of length *string length* would produce a string that is too long.

Explanation: The result of a concatenation must not have a length greater than the maximum allowed for a string.

IBM3813I S Result of *BUILTIN name* applied *repetition value* times to a string of length *string length* would produce a string that is too long.

Explanation: The result of COPY and REPEAT must not have a length greater than the maximum allowed for a string.

IBM3814I S Unsupported use of aggregate expression.

Explanation: The only valid aggregate expression is the use of an array name as the first argument to the HBOUND or LBOUND built-in functions.

IBM3815I S Operand in bit operation must have length less than 32768.

Explanation: Bit operations are limited to strings of length 32767 or less.

IBM3816I S Second and third arguments to the TRANSLATE built-in function must have length less than 32768.

Explanation: The TRANSLATE built-in function is not supported if the second or third argument is longer than 32767 characters.

IBM3817I S Result of *BUILTIN name* would exceed maximum string length.

Explanation: The result of a COMMENT or QUOTE built-in function must not be a string that would have length greater than the supported maximum.

IBM3820I S Under the INCONLY option, the use of INCLUDE or XINCLUDE as a macro procedure name is invalid unless the colon follows immediately after the name.

Explanation: If you must use INCLUDE or XINCLUDE as a macro name, put the colon on the same line as the name.

IBM3821I S Under the INCONLY option, the use of INCLUDE or XINCLUDE as a macro statement label is invalid unless the colon follows immediately after the name.

Explanation: If you must use INCLUDE or XINCLUDE as a macro statement label, put the colon on the same line as the name.

IBM3822I S Under the INCONLY option, the use of INCLUDE or XINCLUDE as a macro variable that is the target of an assignment is invalid unless the equals sign follows immediately after the name.

Explanation: If you must use INCLUDE or XINCLUDE as a macro variable name, put the equals sign in the assignment on the same line as the name. For example, change the first assignment below into the second.

```
%xinclude
  = 17;

%xinclude = 17;
```

IBM3837I S GOTO target is inside a (different) DO loop.

Explanation: The target of a GOTO cannot be inside a DO loop unless the GOTO itself is in the same DO loop.

IBM3841I S The INCLUDE file *include-file-name* could not be opened.

Explanation: The INCLUDE file could not be found, or if found, it could not be opened.

IBM3842I S Statements are nested too deep.

Explanation: The nesting of PROCEDURE, DO, SELECT and similar statements is greater than that supported by the compiler. Rewrite the program so that it is less complicated.

IBM3844I S The *function name* built-in is not supported.

Explanation: Support for the indicated built-in function has been discontinued.

IBM3846I S The *keyword* statement is not supported.

Explanation: Support for the indicated statement has been discontinued.

IBM3848I S Use of iSUB is not supported.

Explanation: iSUB is only supported in syntax checking.

IBM3849I S *type type* type *type name* is not defined.

Explanation: If TYPE x is used in a declaration, x must be a defined type.

IBM3850I S TYPEs must be defined before their use.

Explanation: The DEFINE STRUCTURE or DEFINE ALIAS statement for a type x must precede any of use of x as attribute type. The following two statements should be in the opposite order.

```
dcl x type point;

define structure
```

```
  1 point
    2 x  fixed bin(31),
    2 y  fixed bin(31);
```

IBM3851I S INITIAL values for *type type* type *type name* must be in increasing order.

Explanation: Any values specified in INITIAL clauses in an ORDINAL definition must be in strictly increasing order.

IBM3852I S INITIAL values for *type type* type *type name* must be less than 2G.

Explanation: ORDINAL values must fit in the range of a FIXED BIN(31) variable.

IBM3853I S Nesting of DO statements exceeds the maximum.

Explanation: DO statements can be nested only 100 deep. Simplify the program.

IBM3854I S Nesting of IF statements exceeds the maximum.

Explanation: IF statements can be nested only 100 deep. Simplify the program.

IBM3855I S Nesting of SELECT statements exceeds the maximum.

Explanation: SELECT statements can be nested only 50 deep. Simplify the program.

IBM3856I S Nesting of blocks exceeds the maximum.

Explanation: Blocks must be nested only 30 deep.

IBM3857I S Only one description is allowed in a structure definition.

Explanation: The syntax allows the name in a structure definition to be followed by a description list, but that description list must consist of exactly one structure description. The following is invalid:

```
define structure
  1 point
    2 x  fixed bin(31),
    2 y  fixed bin(31),
  1 rectangle
    2 upper_left   type point,
    2 lower_right  type point;
```

IBM3858I S All the names in the ORDINAL *ordinal-name* have been previously declared.

Explanation: None of the names in an ORDINAL should have been declared elsewhere. If they are, perhaps the ORDINAL definition has been accidentally repeated.

IBM3859I S Storage attributes are invalid in structure definition.

Explanation: Storage attributes, such as AUTOMATIC and BYADDR, must be specified with variables declared with structure type.

IBM3860I S DEFINE STRUCTURE may not specify an array of structures.

Explanation: The level 1 name in a structure definition may not have the DIMENSION attribute.

IBM3861I S Open of dbrm dataset failed.

Explanation: The open of the .dbrm dataset to be used by the SQL preprocessor failed. A possible cause might be lack of write authoriy to the compile directory.

IBM3862I S Dynamic allocation of DBRMLIB failed with the SVC 99 info code *info-code* and the SVC 99 error code *error-code* .

Explanation: The dynamic allocation of the DBRMLIB failed with the indicated SVC 99 info and error codes.

IBM3863I S The DBRMLIB compiler option must be specified.

Explanation: In order to perform a compile using the SQL preprocessor without the INCONLY option, your must specify the DBRMLIB compiler option.

IBM3870I S The FETCH of the CICS backend failed.

Explanation: Check that the CICS modules are accessible, otherwise report this error to IBM.

IBM3871I S The CICS backend reported an internal error while attempting to perform its initialization.

Explanation: Report this error to IBM.

IBM3872I S The CICS backend reported an internal error while attempting to parse its options.

Explanation: Report this error to IBM.

IBM3873I S **The CICS backend reported an internal error while attempting to build and emit the local declares.**

Explanation: Report this error to IBM.

IBM3874I S **The CICS backend reported an internal error while attempting to translate an EXEC statement.**

Explanation: Report this error to IBM.

IBM3875I S **The CICS backend reported an internal error while attempting to translate a CICS macro (such as DFHVALUE).**

Explanation: Report this error to IBM.

IBM3876I S **The CICS backend reported an internal error while attempting to perform its termination.**

Explanation: Report this error to IBM.

IBM3877I S **The SQL backend reported an internal error while attempting to perform its initialization.**

Explanation: Report this error to IBM.

IBM3878I S **SQL initialization did not complete successfully.**

Explanation: See the additional messages produced by the SQL backend.

IBM3880I S **The reference** *reference* **could not be resolved.**

Explanation: All SQL host variables must be declared within the current block scope.

IBM3881I S **The reference** *reference* **is ambiguous.**

Explanation: All SQL host variables must be unambiguous. This can be fixed by supplying enough structure qualification.

IBM3882I S **The indicator array** *reference* **must have only one dimension.**

Explanation: An indicator array in an EXEC SQL statement must not be multi-dimensional.

IBM3883I S **The indicator array** *reference* **must have constant bounds.**

Explanation: An indicator array in an EXEC SQL statement must have bounds that are specified simply as optionally signed integers.

IBM3884I S **The indicator variable** *reference* **is used with a structure and hence must be an array or a structure.**

Explanation: An indicator variable for a structure in an EXEC SQL statement must be an array or a structure.

IBM3885I S **The host variable** *host-variable* **must have only one dimension.**

Explanation: A host variable in an EXEC SQL statement must not be multi-dimensional.

IBM3886I S **The host variable** *host-variable* **must have constant bounds.**

Explanation: A host variable in an EXEC SQL statement must have bounds that are specified simply as optionally signed integers.

IBM3887I S **The host variable** *host-variable* **must be CONNECTED.**

Explanation: A host variable in an EXEC SQL statement must be one-dimensional and that dimension must not be specified on a parent unless the parent has the DIMACROSS attribute.

IBM3888I S **The reference** *host-reference* **has no corresponding DB2 type.**

Explanation: All SQL host variables must have a corresponding DB2 type. For example, while FIXED DEC(7,-2) is valid in a PL/I declaration, there is no corresponding DB2 type because DB2 requires that in FIXED DEC(p,q), q is non-negative and no greater than p.

IBM3889I S **The reference** *host-reference* **is a union and thus must not be used as a host variable.**

Explanation: All SQL host variables must have a corresponding DB2 type. There is no type matching a union.

IBM3890I S **The reference** *host-reference* **is an array of structures and thus must not be used as a host variable.**

Explanation: A structure may be used as a host variable only if it is not an array.

IBM3891I S **Since the structure reference** *host-reference* **contains an array, it must not have an indicator that is a scalar or an array of scalars.**

Explanation: A structure containing an array may be

used as a host variable with an indicator variable only if that indicator variable is a similar structure.

IBM3892I S **The reference** *host-reference* **contains a substructure and thus must not be used as a host variable.**

Explanation: A structure may be used as a host variable only if none of its members are structures.

IBM3893I S **The reference** *host-reference* **contains unnamed elements and thus must not be used as a host variable.**

Explanation: A structure may be used as a host variable only if all of its members are named.

IBM3894I S **The indicator variable** *reference* **must be FIXED BIN(15).**

Explanation: An indicator variable must be a native, real halfword integer.

IBM3895I S **The indicator variable** *reference* **is used with an array and hence must be an array as well.**

Explanation: An indicator variable in an EXEC SQL statement must be an array if it is used with an array.

IBM3896I S **The VALUE reference** *host-reference* **could not be reduced to a character literal and thus must not be used as a host variable.**

Explanation: A reference with the VALUE attribute may be used as a host variable with the SQL characterl type if it can be reduced to a CHARACTER literal. See the Programming Guide for more details.

IBM3897I S **The VALUE reference** *host-reference* **could not be reduced to a numeric literal and thus must not be used as a host variable.**

Explanation: A reference with the VALUE attribute may be used as a host variable with the SQL integer or decimal type if it can be reduced to a REAL FIXED literal. See the Programming Guide for more details.

IBM3898I S **The VALUE reference** *host-reference* **does not have character, integer or decimal type and thus must not be used as a host variable.**

Explanation: A reference with the VALUE attribute may be used as a host variable only if it has a SQL type of character, integer or decimal.

IBM3899I S **The reference** *reference name* **is ambiguous.**

Explanation: Enough qualification must be provided to make any reference unique.

IBM3900I S **The dot-qualified reference** *reference name* **is unknown.**

Explanation: The named reference is not a member of any structure or union declared in the block in which it is referenced or declared in any block containing that block.

IBM3901I S **The element** *reference name* **in the indicator structure must have the same array bounds as the corresponding element in the host structure.**

Explanation: In :x:y, if x and y are both structures, then for any element of y that is an array, the corresponding element of x must be an array with the same bounds and vice versa.

IBM3902I S **Argument to the** *BUILTIN name* **built-in must be a structure.**

Explanation: The argument to the named built-in subroutine must be a structure.

IBM3903I S **The indicator** *reference name* **must not be a uinon.**

Explanation: In :x:y, y must not be a union.

IBM3909I S **The** *attribute* **attribute conflicts with the** *attribute* **attribute.**

Explanation: The named attributes, for example PARAMETER and INITIAL, are mutually exclusive.

IBM3911I S **The statement label** *identifier* **has already been declared.**

Explanation: All statement labels in any block must be unique.

IBM3914I S **GOTO target must be a LABEL reference.**

Explanation: x in GOTO x must have type LABEL. x must not have type FORMAT.

IBM3915I S **GOTO target must be a scalar.**

Explanation: x in GOTO x must not be an array.

IBM3916I S The procedure *proc-name* has already been defined.

Explanation: Sister procedures must have different names.

```
% b: proc;
% end;
% b: proc;
% end;
```

IBM3917I S Program contains no valid source lines.

Explanation: The source contains either no statements or all statements that it contains are invalid.

IBM3920I S FIXED BINARY constant contains too many digits.

Explanation: A FIXED BINARY constant must contain 31 or fewer digits.

IBM3921I S FIXED DECIMAL constant contains too many significant digits.

Explanation: The maximum precision of FIXED DECIMAL constants is set by the FIXEDDEC suboption of the LIMITS compiler option.

IBM3922I S Exponent in FLOAT BINARY constant contains more digits than the implementation maximum.

Explanation: The exponent in a FLOAT BINARY constant may contain no more than 5 digits.

IBM3923I S Mantissa in FLOAT BINARY constant contains more significant digits than the implementation maximum.

Explanation: The mantissa in a FLOAT BINARY constant may contain no more than 64 digits.

IBM3924I S Exponent in FLOAT DECIMAL constant contains more digits than the implementation maximum.

Explanation: The exponent in a FLOAT BINARY constant may contain no more than 4 digits.

IBM3925I S Mantissa in FLOAT DECIMAL constant contains more significant digits than the implementation maximum.

Explanation: The mantissa in a FLOAT BINARY constant may contain no more than 18 digits.

IBM3926I S Constants must not exceed 30720 bytes.

Explanation: The number of bytes used to represent a constant in your program must not exceed 30720. This limit holds even for bit strings where the internal representation will consume only one-eighth the number of bytes as the external representation does.

IBM3927I S Numeric constants must be real, unscaled and fixed.

Explanation: Any complex, scaled or floating point constant will be converted to an integer value.

```
%a = 3.1415;
```

IBM3928I S Only B, BX and X string suffixes are supported.

Explanation: G, GX, M, A and E string suffixes are not supported.

```
%a = '31'e;
```

IBM3929I S EXEC SQL statement must be in a PROCEDURE.

Explanation: The only EXEC SQL statements allowed at the PACKAGE level are EXEC SQL BEGIN DECLARE SECTION, EXEC SQL END DECLARE SECTION, nonexecutable EXEC SQL DECLARE, and EXEC SQL INCLUDE other than EXEC SQL INCLUDE SQLCA and EXEC SQL INCLUDE SQLDA.

IBM3930I S Invalid syntax in statement-form of procedure invocation. Text up to next semicolon will be ignored.

Explanation: In the invocation of a statement-form procedure, all characters that are not part of comments or key names should be enclosed in parentheses following one of the keys. For example, the "+" in the display statement below should not be present.

```
%a: proc( x ) stmt returns( char );
  dcl x char;
  return( 1729 );
%end;
%act a;

display( a + x(5); );
```

IBM3931I S Under the FIXED(DEC) option, decimal constants must have no more than 5 digits.

Explanation: Under the FIXED(BIN), decimal

constants that represent any valid FIXED BIN(31) number are supported.

IBM3934I S EXEC SQL INCLUDE statement has incorrect syntax.

Explanation: EXEC SQL INCLUDE must be followed by one identifier and then by a semicolon.

IBM3935I S The FETCH of the SQL backend failed.

Explanation: Check that the SQL modules are accessible, otherwise report this error to IBM.

IBM3936I S The SQL backend must be from DB2 V9 or later.

Explanation: Switch to a more current level of DB2.

IBM3937I S The EXEC SQL statement is too long.

Explanation: The EXEC SQL statement must be less than 500K bytes long.

IBM3938I S The EXEC SQL statement has too many host variables

Explanation: The EXEC SQL statement must use no more than 1500 host variables.

IBM3939I S The DBNAME option must specify a valid database name.

Explanation: When invoking the SQL preprocessor on Windows or AIX, the DBNAME option must be specified, and the option must specify a valid database name.

IBM3943I S The number of error messages allowed by the FLAG option has been exceeded.

Explanation: Compilation will terminate when the number of messages has exceeded the limit set in the FLAG compiler option.

IBM3948I S *condition-name* **condition with ONCODE=***oncode-value* **raised while evaluating expression.**

Explanation: Evaluation of an expression raised the named condition.

```
%a = a / 0;
```

IBM3949I S Parameter name *identifier* **appears more than once in parameter list.**

Explanation: Each identifier in a parameter list must be unique.

```
a: proc( b, c, b );
```

IBM3950I S An asterisk iteration factor can be applied only to the last expression in the INITIAL item list for *variable-name***.**

Explanation: Since an asterisk iteration factor completes the initialization of a variable, it cannot be followed by more initial values.

```
%dcl a(10)  fixed init( 1, 2, (*) 0, 8 );
```

IBM3951I S An asterisk iteration factor cannot be used in the nested INITIAL item list for *variable-name***.**

Explanation: An asterisk iteration can be used only in a non-nested INITIAL item list. The following example is invalid.

```
%dcl a(20) fixed init( (2) ( 1, (*) 2 ) );
```

IBM3952I S INITIAL attribute on the parameter *variable-name* **is invalid.**

Explanation: A parameter cannot have an INITIAL attribute.

IBM3953I S INITIAL list contains *count* **items, but the array** *variable name* **contains only** *array size***. Excess is ignored.**

Explanation: For an array, an INITIAL list should not contain more values than the array has elements.

```
%dcl b(5) init( (10) 0 );
```

IBM3956I S ITERATE is valid only for iterative DO-groups.

Explanation: ITERATE is not valid inside type-I do groups.

IBM3957I S RETURN statement outside of a PROCEDURE is invalid.

Explanation: RETURN statements are valid only inside procedures.

IBM3958I S INCLUDE statement inside of a PROCEDURE is invalid.

Explanation: INCLUDE statements are permitted only outside any preprocessor procedures.

```
%a: proc;
   include sample;
%end;
```

IBM3959I S Length of parameter exceeds 32767 bytes.

Explanation: Parameters to macro procedures must be no longer than 32767 bytes.

IBM3960I S End-of-source has been encountered after an unmatched comment marker.

Explanation: An end-of-comment marker is probably missing.

IBM3961I S End-of-source has been encountered after an unmatched quote.

Explanation: A closing quote is probably missing.

IBM3962I S Replacement value contains no end-of-comment delimiter. A comment delimiter will be assumed at the end of the replacement value.

Explanation: An end-of-comment marker is probably missing.

IBM3963I S Replacement value contains no end-of-string delimiter. A string delimiter will be assumed at the end of the replacement value.

Explanation: A closing quote is probably missing.

IBM3964I S ANSWER statement outside of a PROCEDURE is invalid.

Explanation: ANSWER statements are valid only inside procedures.

IBM3965I S ANSWER statement inside of a PROCEDURE with RETURNS is invalid.

Explanation: ANSWER statements are not valid inside functions.

```
%a: proc returns( char );
  answer( 'this is invalid' );
  return( 'this is ok however' );
%end;

%b: proc;
  answer( 'this is valid' );
%end;
```

IBM3966I S Source has caused too many rescans.

Explanation: A rescan of a replacement string or a rescan of a string returned by a preprocessor has caused further replacement leading to another rescan etc., and the maximum depth of rescanning was exceeded. For instance, the following macro, which is meant to count the number of dcl statements in a compilation, would produce this message. If the %ACTIVATE statement specified NORESCAN, it would work correctly.

```
%dcl dcl_Count fixed;
%dcl_Count = 0;

%dcl: proc returns( char );
   dcl_count = dcl_count + 1;
   return( 'dcl' );
%end;

%activate dcl;
```

IBM3967I S CALL statement outside of a PROCEDURE is invalid.

Explanation: CALL statements are valid only when they are inside macro procedures.

IBM3968I S CALL reference is undefined.

Explanation: CALL reference must be a declared macro procedure.

IBM3969I S CALL reference is not a macro entry.

Explanation: CALL reference must be a declared macro procedure.

IBM3970I S CALL reference must not be a function.

Explanation: A CALL reference must not have the RETURNS attribute.

IBM3971I S CALL reference must not have the STATEMENT option.

Explanation: A CALL reference must not have the STATEMENT option.

IBM3974I S Every shift-in character after the left margin of a source line must have a matching shift-out character before the right margin of the same line.

Explanation: DBCS shift codes must be paired.

IBM3975I S Every shift-in character within a string generated for rescan must have a matching shift-out character within that same string.

Explanation: DBCS shift codes must be paired.

IBM3976I S DBCS characters are allowed only in G and M constants.

Explanation: Hex strings (strings ending in one of the suffixes X, BX, B4, GX or XN), bit strings, (strings ending in the suffix B), and character strings not ending in the suffix M must contain only SBCS characters.

IBM3977I S SBCS characters are not allowed in G constants.

Explanation: Mixed SBCS and DBCS is allowed only in M constants.

IBM3978I S Invalid use of SBCS encoded as DBCS.

Explanation: Outside of comments, SBCS can be encoded as DBCS only as part of an identifier.

IBM3980I S Recursion of procedures is not allowed.

Explanation: A procedure must not invoke itself directly or indirectly.

IBM3981I S *BUILTIN function* **may not be used outside a procedure.**

Explanation: The named built-in function may be used only inside procedures.

IBM3982I S Procedure *procedure-name* **is undefined and cannot be invoked.**

Explanation: A procedure must be defined (correctly) before it can be invoked.

IBM3983I S Premature end-of-source in scan.

Explanation: The source ended during a scan when a right parenthesis or semicolon was required.

```
%a: proc() stmt returns( char );
  return( '1729' );
%end;
%dcl a entry;

a /* and no more source follows */
```

IBM3984I S File *filename* **could not be opened.**

Explanation: The named source file could not be opened. Make sure that the file is named correctly, that it exists and that it is readable.

IBM3985I S Semicolon found before required closing right parenthesis.

Explanation: A statement contained a semicolon before a right parenthesis which is needed to match an earlier left parenthesis in the statement.

```
select( a ; );
```

IBM3986I S IF statement syntax is invalid.

Explanation: A statement that appears to be an IF statement has invalid syntax.

```
if a > 0 ; then
```

IBM3987I S Statement must start with a keyword or assignment target.

Explanation: After any condition prefixes and labels, statements must start with either a keyword or, if the statement is an assignment statement, it must start with an identifier or BIND reference. The flagged statement starts with some other lexical element. This may indicate that a semicolon that is meant for the previous statement is misplaced or that an element of this statement has been erroneously omitted.

```
a =0 b; = a;
```

IBM3988I S Statement has invalid syntax.

Explanation: The flagged statement is not valid PL/I. This may indicate that a semicolon that is meant for the previous statement is misplaced or that an element of this statement has been erroneously omitted.

```
put skip garbage;
```

IBM3993I S Internal preprocessor error: assertion failed on line *source line* **in** *procedure name* **in** *package name*

Explanation: This message indicates that there is an error in the preprocessor. Report the problem to IBM.

IBM3994I S Source is not valid UTF-8.

Explanation: The source file contains lines that would be rejected by the UVALID built-in function.

IBM3995I S Generated text contains invalid UTF-8.

Explanation: The text produced by an ANSWER or RETURNS statement would be rejected by the UVALID built-in function.

IBM3996I S Internal preprocessor error: protection exception in *module name***.**

Explanation: This message indicates that there is an error in the preprocessor. Report the problem to IBM.

IBM3997I S Internal preprocessor error: no WHEN clause satisfied within *module name***.**

Explanation: This message indicates that there is an error in the preprocessor. Report the problem to IBM.

IBM3998I S *note*

Explanation: This message is used to report DB2 or CICS backend messages with a return code of 16.

IBM3999I U *note*

Explanation: This message is used by %NOTE statements with a return code of 16.

Chapter 6. Code Generation Messages (5000-5999)

IBM5001 INTERNAL COMPILER ERROR: *text*

Explanation: An internal compiler error occurred during compilation.

Contact your Service Representative.

IBM5002 Virtual storage exceeded.

Explanation: The compiler ran out of memory trying to compile the file. This sometimes happens with large files or programs with large functions. Note that very large programs limit the amount of optimization that can be done.

Shut down any large processes that are running, ensure your swap path is large enough, turn off optimization, and redefine your virtual storage to a larger size. You can also divide the file into several small sections or shorten the function.

IBM5003 *text*

Explanation: General error message.

IBM5031 Unable to open file *filename***.**

Explanation: The compiler could not open the specified file.

Ensure the file name is correct. Ensure that the correct file is specified. If the file is located on a LAN drive, ensure the LAN is working properly. Also, the file may be locked by another process or access may be denied because of insufficient permission.

IBM5032 An error occurred while reading file *filename***.**

Explanation: The compiler detected an error while reading from the specified file.

Ensure that the correct file is being read and has not been damaged. If the file is located on a LAN drive, ensure the LAN is working properly.

IBM5033 An error occurred while writing to file *filename***.**

Explanation: The compiler detected an error while writing to the specified file.

Ensure that the correct file is specified. If the file is located on a LAN drive, ensure the LAN is working properly.

IBM5034 Read-only pointer initialization of dynamically allocated object *name* **is not valid.**

Explanation: The value of a read-only pointer must be known at compile time; a pointer cannot be read-only and point to a dynamically allocated object at the same time because the address of the pointee is known at run time only.

Modify the code so that the pointer is initialized with a read-only value or make the pointer read-write.

IBM5051 Function *function-name* **exceeds size limit.**

Explanation: The ACU for the function exceeds the LIMIT specified in the INLINE suboption.

Increase LIMIT if feasible to do so.

IBM5052 Function *function-name* **is (or grows) too large to be inlined.**

Explanation: A function is too large to be inlined into another function.

IBM5053 Some calls to function *function-name* **cannot be inlined.**

Explanation: At least one call is either directly recursive, or the wrong number of parameters were specified.

Check all calls to the function specified and make that number of parameters match the function definition.

IBM5054 Automatic storage for function *function-name* **increased to over** *value***.**

Explanation: The size of automatic storage for function increased by at least 4 KB due to inlining.

Avoid inlining of functions which have large automatic storage.

IBM5055 Parameter area overflow while compiling *function-name***. Parameter area size exceeds the allowable limit of** *value***.**

Explanation: The parameter area for a function resides in the first 4K of automatic storage for that function. This message indicates that the parameter area cannot fit into 4K.

Reduce the size of the parameter area by passing fewer parameters or by passing the address of a large structure rather than the structure itself.

© Copyright IBM Corp. 1999, 2018

IBM5057 ***name*** **section size cannot exceed 16777215 bytes. Total section size is** ***value*** **bytes.**

Explanation: A Data or Code section cannot exceed 16M in size.

Partition input source files into multiple source files which can be compiled separately.

IBM5101 **Maximum spill size of** ***value*** **is exceeded in function** ***function-name*****.**

Explanation: Spill size is the size of the spill area. Spill area is the storage allocated if the number of machine registers is not sufficient for program translation.

Reduce the complexity of the program and recompile.

IBM5102 **Spill size for function** ***function-name*** **is not sufficient. Recompile specifying option SPILL(n) where** ***lower-limit*** **< n <=** ***upper-limit*****.**

Explanation: Spill size is the size of the spill area. Spill area is the storage allocated if the number of machine registers is not sufficient for program translation.

Recompile using the SPILL(n) option *lower-limit* < n <= *upper-limit* or with a different OPT level.

IBM5103 **Internal error while compiling function** ***function-nametext*****.**

Explanation: An internal compiler error occurred during compilation.

Contact your Service Representative or compile with a different OPT level.

IBM5104 **Internal error while compiling function** ***function-name text*****. Compilation terminated.**

Explanation: An internal compiler error of high severity has occurred.

Contact your Service Representative. Be prepared to quote the text of this message.

IBM5105 **Constant table overflow compiling function** ***function-name*****. Compilation terminated.**

Explanation: The constant table is the table that stores all the integer and floating point constants.

Reduce the number of constants in the program and recompile.

IBM5106 **Instruction in function** ***function-name*** **on line** ***value*** **is too complex. Compilation terminated.**

Explanation: The specified instruction is too complex to be optimized.

Reduce the complexity of the instruction and recompile, or recompile with a different OPT level.

IBM5107 **Program too complex in function** ***function-name*****.**

Explanation: The specified function is too complex to be optimized.

Reduce the complexity of the program and recompile, or recompile with a different OPT level.

IBM5108 **Expression too complex in function** ***function-name*****. Some optimizations not performed.**

Explanation: The specified expression is too complex to be optimized.

Reduce the complexity of the expression or compile with a different OPT level.

IBM5109 **Infinite loop detected in function** ***function-name*****. Program may not stop.**

Explanation: A loop which may be infinite has been detected in the given function, and your code may need to be changed. However, sometimes the compiler will issue this message when your code is OK. For example, if the loop is exited via a GOTO out of an ON-unit, the compiler may issue this message although you would not need to change your code.

Recode the loop so that it will end.

IBM5110 **Loop too complex in function** ***function-name*****. Some optimizations not performed.**

Explanation: The specified loop is too complex to be optimized.

No action is required.

IBM5111 **Division by zero detected in function** ***function-name*****. Runtime exception may occur.**

Explanation: A division by zero has been detected in the given function.

Recode the expression to eliminate the divide by zero.

IBM5112 **Exponent is non-positive with zero as base in function** *function-name***. Runtime exception may occur.**

Explanation: This is a possible floating-point divide by zero.

Recode the expression to eliminate the divide by zero.

IBM5113 **Unsigned division by zero detected in function** *function-name***. Runtime exception may occur.**

Explanation: A division by zero has been detected in the given function.

Recode the expression to eliminate the divide by zero.

IBM5114 **Internal error while compiling function** *function-name text***.**

Explanation: An internal compiler error of low severity has occurred.

Contact your Service Representative or compile with a different OPT level.

IBM5115 **Control flow too complex in function** *function-name* **; number of basic blocks or edges exceeds** *value***.**

Explanation: Basic blocks are segments of executable code without control flow. Edges are the possible paths of control flow between basic blocks.

Reduce the complexity of the program and recompile.

IBM5116 **Too many expressions in function** *function-name* **; number of symbolic registers exceeds** *value***.**

Explanation: Symbolic registers are the internal representation of the results of computations.

Reduce the complexity of the program and recompile.

IBM5117 **Too many expressions in function** *function-name***; number of computation table entries exceeds** *value***.**

Explanation: The computation table contains all instructions generated in the translation of a program.

Reduce the complexity of the program and recompile.

IBM5118 **Too many instructions in function** *function-name***; number of procedure list entries exceeds** *value***.**

Explanation: The procedure list is the list of all instructions generated by the translation of each subprogram.

Reduce the complexity of the program and recompile.

IBM5119 **Number of labels in function** *function-name* **exceeds** *value***.**

Explanation: Labels are used whenever the execution path of the program could change; for example: if statements, switch statements, loops or conditional expressions.

Reduce the complexity of the program and recompile.

IBM5120 **Too many symbols in function** *function-name* **; number of dictionary entries exceeds** *value***.**

Explanation: Dictionary entries are used for variables, aggregate members, string literals, pointer dereferences, function names and internal compiler symbols.

Compile the program at a lower level of optimization or simplify the program by reducing the number of variables or expressions.

IBM5121 **Program is too complex in function** *function-name***. Specify MAXMEM option value greater than** *value***.**

Explanation: Some optimizations not performed.

Recompile specifying option MAXMEM with the suggested value for additional optimization.

IBM5122 **Parameter area overflow while compiling** *name***. Parameter area size exceeds** *value***.**

Explanation: The parameter area is used to pass parameters when calling functions. Its size depends on the number of reference parameters, the number and size of value parameters, and on the linkage used.

Reduce the size of the parameter area by passing fewer parameters or by passing the address of a large structure rather than the structure itself.

IBM5123 **Spill size for function** *function-name* **is exceeded. Recompile specifying option SPILL(n) where** *lower-limit* **< n <=** *upper-limit* **for faster spill code.**

Explanation: Spill size is the reserved size of the primary spill area. Spill area is the storage allocated if the number of machine registers is not sufficient for program translation.

Recompile using the SPILL(n) option with *lower-limit* < n <= *upper-limit* for improved spill code generation.

IBM5130 **An error occurred while opening file** *filename***.**

Explanation: The compiler could not open the specified file.

Ensure the file name is correct. Ensure that the correct file is being opened and has not been damaged. If the file is located on a LAN drive, ensure the LAN is working properly. Also, the file may be locked by another process or access may be denied because of insufficient permission.

IBM5131 **An error occurred while writing file** *filename***.**

Explanation: The compiler could not read from the specified file.

Ensure the file name is correct. Ensure that the correct file is being written to and has not been damaged. If the file is located on a LAN drive, ensure the LAN is working properly. Also, the file may be locked by another process or access may be denied because of insufficient permission.

IBM5132 **An error occurred while closing file** *filename***.**

Explanation: The compiler could not write to the specified file.

Ensure the file name is correct. Ensure that the correct file is being closed and has not been damaged. If the file is located on a LAN drive, ensure the LAN is working properly. Also, the file may be locked by another process or access may be denied because of insufficient permission.

IBM5141 **Automatic area for** *function-name* **is too large**

Explanation: Automatic data resides in the stack; the stack size is limited by the target machine addressabilty.

Avoid large structures and large arrays as local variables; try using dynamically allocated data. Alternatively, try to break down the procedure into several smaller procedures.

Chapter 7. Condition codes

Condition codes listed in this section reflect an aggregate of condition codes generated by all implementations. Some might not be generated for a particular platform.

A summary of all condition codes are listed in numerical sequence as follows.

Condition codes 1 through 500

3 This condition is raised if, in a SELECT group, no *WHEN* clause is selected and no *OTHERWISE* clause is present.

4 SIGNAL FINISH, or STOP statement executed.

9 SIGNAL ERROR statement executed.

10 SIGNAL NAME statement executed.

20 SIGNAL RECORD statement executed.

21 Record variable smaller than record size. Either:

- The record is larger than the variable in a READ INTO statement; the remainder of the record is lost.
- The record length specified for a file with fixed-length records is larger than the variable in a WRITE, REWRITE, or LOCATE statement; the remainder of the record is undefined. If the variable is a varying-length string, RECORD is not raised if the SCALARVARYING option is applied to the file.

22 Record variable larger than record size. Either:

- The record length specified for a file with fixed-length records is smaller than the variable in a READ INTO statement; the remainder of the variable is undefined. If the variable is a varying-length string, RECORD is not raised if the SCALARVARYING option is applied to the file.
- The maximum record length is smaller than the variable in a WRITE, REWRITE, or LOCATE statement. For WRITE or REWRITE, the remainder of the variable is lost; for LOCATE, the variable is not transmitted.
- The variable in a WRITE or REWRITE statement indicates a zero length; no transmission occurs. If the variable is a varying-length string, RECORD is not raised if the SCALARVARYING option is applied to the file.

23 Record variable length is either zero or too short to contain the embedded key.

The variable in a WRITE or REWRITE statement is too short to contain the data set embedded key; no transmission occurs. (This case currently applies only to indexed key-sequenced data sets.)

24 Zero length record was read from a REGIONAL data set.

40 SIGNAL TRANSMIT statement executed.

41 Uncorrectable transmission error in output data set.

© Copyright IBM Corp. 1999, 2018

42 Uncorrectable transmission error in input data set.

43 Uncorrectable transmission error on output to index set.

44 Uncorrectable transmission error on input from index set.

45 Uncorrectable transmission error on output to indexed consecutive data set.

46 Uncorrectable transmission error on input from consecutive data set.

50 SIGNAL KEY statement executed.

51 Key specified cannot be found.

52 Attempt to add keyed record that has same key as a record already present in data set; or, in a REGIONAL(1) data set, attempt to write into a region already containing a record.

53 Value of expression specified in KEYFROM option during sequential creation of INDEXED or REGIONAL data set is less than value of previously specified key or region number.

54 Key conversion error, possibly due to region number not being numeric character.

55 Key specification is null string or begins with (8)'1'B or a change of embedded key has occurred on a sequential REWRITE[FROM] for an INDEXED or key-sequenced data set.

56 Attempt to access a record using a key that is outside the data set limits.

57 No space available to add a keyed record on INDEXED insert.

58 Key of record to be added lies outside the range(s) specified for the data set.

70 SIGNAL ENDFILE statement executed.

80 SIGNAL UNDEFINEDFILE statement executed.

81 Conflict in file attributes exists at open time between attributes in DECLARE statement and those in explicit or implicit OPEN statement.

82 Conflict between file attributes and physical organization of data set (for example, between file organization and device type), or indexed data set has not been loaded.

83 After merging ENVIRONMENT options with DD statement and data set label, data set specification is incomplete; for example, block size or record format has not been specified.

84 No DD statement associating file with a data set.

85 During initialization of a DIRECT OUTPUT file associated with a REGIONAL data set, an input/output error occurred.

86 LINESIZE greater than implementation-defined maximum, or invalid value in an ENVIRONMENT option.

87 After merging ENVIRONMENT options with DD statement and data set label, conflicts exist in data set specification; the value of LRECL, BLKSIZE or RECSIZE are incompatible with one another or the DCB FUNCTION specified.

88 After merging ENVIRONMENT options with DD statement and data set

label, conflicts exist in data set specification; the resulting combination of MODE/FUNCTION and record format are invalid.

89 Password invalid or not specified.

90 SIGNAL ENDPAGE statement executed.

91 ENVIRONMENT option invalid for file accessing indexed data set.

92 The requested data set was not available.

93 Error detected by the operating system while opening a data set.

Subcode1	Meaning
50	A nonexistent ISAM file is being opened for input.
51	An unexpected error occurred when opening an ISAM file. Subcode2 gives the return code from ISAM.
52, 53	An unexpected error occurred when opening a native or REGIONAL(1) file.
54	A nonexistent BTRIEVE file is being opened for input.
55	An unexpected error occurred when opening a BTRIEVE file. Subcode2 gives the return code from BTRIEVE.
56	An unexpected error occurred when opening a DDM file.
57, 58	An unexpected error occurred when opening a DDM sequential, DDM relative or DDM indexed file. Subcode2 gives the return code from DDM.
59	An attempt was made to open a file that was already open.
60	A file of invalid type is being opened. An example of this is opening a VSAM file under z/OS UNIX System Services. VSAM files are not supported under z/OS UNIX System Services.
66	Open of a VSAM file failed. Subcode2 gives the feedback code.
76	A retry attempt at opening an SFS file failed.
79	An SFS file opened for input or update could not be found.
119	An unexpected error occurred during dynamic allocation processing for the file.
120	A parsing error occurred during dynamic allocation processing for the file.
121	An unexpected function was detected during dynamic allocation processing for the file.
122	An unsupported file mode was detected during dynamic allocation processing for the file.
123	The DDNAME could not be located during dynamic allocation processing for the file.

94 REUSE specified for a nonreusable data set.

95 Alternate index specified for an index data set is empty.

96 Incorrect environment variable.

97 VSAM server not available to perform the OPEN.

98 Attempt to position the file at the first record failed.

99 File cannot be opened.

Subcode1	Meaning
1 or 2	The extended attributes (EAs) for an existing REGIONAL(1) file could not be located and no RECCOUNT or RECSIZE values were given via the ENVIRONMENT or SET DD option.
3	A positioning error occurred for a sequential output file.

Subcode1	Meaning
4	TYPE (FIXED) was specified for a native file, but the file size was not a multiple of RECSIZE.
5 or 13	A positioning error occurred for a REGIONAL(1) file.
6 - 12	A positioning error occurred for an output file.
21 - 23	AMTHD(DDM) was specified on the SET DD statement for a file, but the DDM DDLs (DUBRUN and DUBLDM) could not be found or accessed.
24	Incorrect extended attribute on a DDM file.
25	The ORGANIZATION option of the ENVIRONMENT attribute conflicts with the type of data set (DDM or native).
26	Conflicts exist with how the file is being used.
27	A composite key was detected with a keyed-opening.
28 - 30	A new DDM file could not be created.
31	A positioning error occurred for a DDM file.
35	AMTHD(BTRIEVE) was specified on the DD environment variable but the BTRIEVE loadable component (BTRCALLS) could not be found or could not be accessed on the system.
36	Unexpected error occurred when opening a BTRIEVE file.
37	A new BTRIEVE file could not be created.
38	A positioning error occurred for a BTRIEVE file.
40	AMTHD(ISAM) was specified on the DD environment variable but the ISAM non-multithreading loadable components (IBMWS20F and IBMWS20G) or the ISAM multithreading loadable components (IBMWM20F and IBMWM20G) could not be found or could not be accessed on the system.
41	Unexpected error occurred when opening an ISAM file.
42	A new ISAM file could not be created.
43	A positioning error occurred for an ISAM file.
60	A file of invalid type is being opened. An example of this is opening a VSAM file under z/OS UNIX System Services. VSAM files are not supported under z/OS UNIX System Services.
62	Query for file information failed for a VSAM file under MVS batch.
63	A non-VSAM file is being opened as a VSAM file under MVS batch.
64	A VSAM file is being opened with an invalid type (that is, the file is not a KSDS, ESDS or RRDS file).
65	A VSAM file is being opened in a non-MVS batch environment. VSAM files are supported only under MVS batch.
66	Open of a VSAM file failed. Subcode 2 gives the feedback code.
67	A VSAM file is being opened as a non-VSAM file under MVS batch.
68	An invalid VSAM file is being opened.
69	Query for file information failed for a native file under MVS batch.
70	Positioning for a VSAM file failed.
71	A VSAM file is being opened under a non-MVS batch environment.
72	An invalid PL/I file is being opened.
73	The SFS library cannot be loaded.
74	The DCE library cannot be loaded.
75	A new SFS file could not be created.
77	Positioning for an SFS file failed.
78	Not enough storage below the line.
80	There was an error processing an empty VSAM file opened for update. Oncode 82 should have been issued.

110 The specified data set or path name could not be found during dynamic allocation processing for the file.

111 An invalid keyword was encountered in the environment variable string during dynamic allocation processing for the file.

112 Conflicting keywords were detected during dynamic allocation processing for the file.

113 A bad delimiter was detected during dynamic allocation processing for the file.

115 The DSN parameter of the environment variable specified a temporary data set name, which is not supported for dynamic allocation.

116 The PATH parameter of the environment variable did not specify an absolute path name.

117 The data set name specified in the DSN keyword of the environment variable was invalid.

118 The member name specified in the DSN keyword of the environment variable was invalid.

119 The path name specified in the PATH keyword of the environment variable was invalid.

120 An error occurred during the dynamic allocation phase for the file associated with the ddname.

121 An error occurred while attempting to dynamically deallocate the file associated with the ddname.

150 SIGNAL STRINGSIZE statement executed or STRINGSIZE condition occurred.

151 Truncation occurred during assignment of a mixed character string.

290 SIGNAL INVALIDOP statement was executed or INVALIDOP exception occurred.

300 SIGNAL OVERFLOW statement executed or OVERFLOW condition occurred.

310 SIGNAL FIXEDOVERFLOW statement executed or FIXEDOVERFLOW condition occurred.

320 SIGNAL ZERODIVIDE statement executed or ZERODIVIDE condition occurred.

330 SIGNAL UNDERFLOW statement executed or UNDERFLOW condition occurred.

340 SIGNAL SIZE statement executed; or high-order nonzero digits have been lost in an assignment to a variable or temporary, or significant digits have been lost in an input/output operation.

341 High order nonzero digits have been lost in an input/output operation.

350 SIGNAL STRINGRANGE statement executed or STRINGRANGE condition occurred.

360 Attempt to allocate a based variable within an area that contains insufficient free storage for allocation to be made.

361 Insufficient space in target area for assignment of source area.

362 SIGNAL AREA statement executed.

400 SIGNAL ATTENTION statement executed.

430 SIGNAL ASSERTION.

431 An ASSERT TRUE/FALSE statement without a TEXT clause failed.

432 An ASSERT TRUE/FALSE statement with a TEXT clause failed .

433 An ASSERT UNREACHABLE statement without a TEXT clause failed.

434 An ASSERT UNREACHABLE statement with a TEXT clause failed.

435 An ASSERT COMPARE statement without a TEXT clause failed.

436 An ASSERT COMPARE statement with a TEXT clause failed.

450 SIGNAL STORAGE statement executed.

451 ALLOCATE statement or ALLOCATE built-in function failed; insufficient storage to satisfy request.

500 SIGNAL CONDITION (name) statement executed.

Condition codes 501 through 1000

520 SIGNAL SUBSCRIPTRANGE statement executed, or subscript has been evaluated and found to lie outside its specified bounds.

600 SIGNAL CONVERSION statement executed.

601 Invalid conversion attempted during input/output of a character string.

603 Error during processing of an F-format item for a GET STRING statement.

604 Error during processing of an F-format item for a GET FILE statement.

605 Error during processing of an F-format item for a GET FILE statement following a TRANSMIT condition.

606 Error during processing of an E-format item for a GET STRING statement.

607 Error during processing of an E-format item for a GET FILE statement.

608 Error during processing of an E-format item for a GET FILE statement following a TRANSMIT condition.

609 Error during processing of a B-format item for a GET STRING statement.

610 Error during processing of a B-format item for a GET FILE statement.

611 Error during processing of a B-format item for a GET FILE statement following TRANSMIT condition.

612 Error during character value to arithmetic conversion.

613 Error during character value to arithmetic conversion for a GET or PUT FILE statement.

614 Error during character value to arithmetic conversion for a GET or PUT FILE statement following a TRANSMIT condition.

615 Error during character value to bit value conversion.

616 Error during character value to bit value conversion for a GET or PUT FILE statement.

617 Error during character value to bit value conversion for a GET or PUT FILE statement following a TRANSMIT condition.

618 Error during character value to picture conversion.

619 Error during character value to picture conversion for a GET or PUT FILE statement.

620 Error during character value to picture conversion for a GET or PUT FILE statement following a TRANSMIT condition.

621 Error in decimal P-format item for a GET STRING statement.

622 Error in decimal P-format input for a GET FILE statement.

623 Error in decimal P-format input for a GET FILE statement following a TRANSMIT condition.

624 Error in character P-format input for a GET FILE statement.

625 Error exists in character P-format input for a GET FILE statement.

626 Error exists in character P-format input for a GET FILE statement following a TRANSMIT condition.

627 A graphic or mixed character string encountered in a nongraphic environment.

628 A graphic or mixed character string encountered in a nongraphic environment on input.

629 A graphic or mixed character string encountered in a nongraphic environment on input after TRANSMIT was detected.

633 An invalid character detected in a *X*, *BX*, or *GX* string constant.

634 An invalid character detected in a *X*, *BX*, or *GX* string constant on input.

635 An invalid character detected in a *X*, *BX*, or *GX* string constant on input after *TRANSMIT* was detected.

640 Conversion from picture contained an invalid character.

641 Conversion from picture contained an invalid character on input or output.

642 Conversion from picture contained an invalid character on input after TRANSMIT was detected.

643 Error during processing of a graphic F-format item for a GET STRING statement.

644 Error during processing of a graphic F-format item for a GET FILE statement.

645 Error during processing of a graphic F-format item for a GET FILE statement following a TRANSMIT condition.

646 Error during processing of a graphic E-format item for a GET STRING statement.

647 Error during processing of a graphic E-format item for a GET FILE statement.

648 Error during processing of a graphic E-format item for a GET FILE statement following a TRANSMIT condition.

649 Error during processing of a graphic B-format item for a GET STRING statement.

650 Error during processing of a graphic B-format item for a GET FILE statement.

651 Error during processing of a graphic B-format item for a GET FILE statement following TRANSMIT condition.

652 Error during graphic character value to arithmetic conversion.

653 Error during graphic character value to arithmetic conversion for a GET or PUT FILE statement.

654 Error during graphic character value to arithmetic conversion for a GET or PUT FILE statement following a TRANSMIT condition.

655 Error during graphic character value to bit value conversion.

656 Error during graphic character value to bit value conversion for a GET or PUT FILE statement.

657 Error during graphic character value to bit value conversion for a GET or PUT FILE statement following a TRANSMIT condition.

658 Error during graphic character value to picture conversion.

659 Error during graphic character value to picture conversion for a GET or PUT FILE statement.

660 Error during graphic character value to picture conversion for a GET or PUT FILE statement following a TRANSMIT condition.

661 Error in decimal graphic P-format item for a GET STRING statement.

662 Error in decimal graphic P-format input for a GET FILE statement.

663 Error in decimal graphic P-format input for a GET FILE statement following a TRANSMIT condition.

664 Error in character graphic P-format input for a GET FILE statement.

665 Error exists in character graphic P-format input for a GET FILE statement.

666 Error exists in character graphic P-format input for a GET FILE statement following a TRANSMIT condition.

667 No SBCS equivalent in the GRAPHIC conversion to character.

668 No SBCS equivalent in the GRAPHIC conversion to character on input.

669 No SBCS equivalent in the GRAPHIC conversion to character on input following a TRANSMIT condition.

670 Unknown source attributes.

671 Unknown source attributes on input.

672 Unknown source attributes on input following a TRANSMIT condition.

673 Error during WIDECHAR value to character conversion.

674 Error during WIDECHAR value to character conversion for a GET or PUT FILE statement.

675 Error during WIDECHAR value to character conversion for a GET or PUT FILE statement following a TRANSMIT condition.

676 Error during WIDECHAR value to arithmetic conversion.

677 Error during WIDECHAR value to arithmetic conversion for a GET or PUT FILE statement.

678 Error during WIDECHAR value to arithmetic conversion for a GET or PUT FILE statement following a TRANSMIT condition.

679 Error during WIDECHAR value to bit value conversion.

680 Error during WIDECHAR value to bit value conversion for a GET or PUT FILE statement.

681 Error during WIDECHAR value to bit value conversion for a GET or PUT FILE statement following a TRANSMIT condition.

682 Error during WIDECHAR value to picture conversion.

683 Error during WIDECHAR value to picture conversion for a GET or PUT FILE statement.

684 Error during WIDECHAR value to picture conversion for a GET or PUT FILE statement following a TRANSMIT condition.

Condition codes 1001 through 1499

1001 EVENT variable already used with a DISPLAY statement.

1002 GET or PUT STRING specifies data exceeding size of string.

1003 Further output prevented by TRANSMIT or KEY conditions previously raised for the data set.

1004 Attempt to use PAGE, LINE, or SKIP <= 0 for nonprintable file.

1005 In a DISPLAY(expression) REPLY (character-reference) statement, expression or character-reference is zero length.

1007 A REWRITE or a DELETE statement not preceded by a READ.

1008 Unrecognized field preceding the assignment symbol in a string specified in a GET STRING DATA statement.

1009 An input/output statement specifies an operation or an option which conflicts with the file attributes.

1010 A built-in function or pseudovariable referenced an unopened file.

1011 Data management detected an input/output error but is unable to provide any information about its cause.

1013 Previous input operation incomplete; REWRITE or DELETE statement specifies data which has been previously read in by a READ statement with an EVENT option, and no corresponding WAIT has been executed.

1014 Attempt to initiate further input/output operation when number of incomplete operations equals number specified by ENVIRONMENT option NCP(n) or by default.

1015 Event variable specified for an input/output operation when already in use.

1016 After UNDEFINEDFILE condition raised as a result of an unsuccessful attempt to implicitly open a file, the file was found unopened on normal return from the ON-unit.

1018 End of file or string encountered in data before end of data-list or in edit-directed transmission format list.

1019 Attempt to close file not opened in current process.

1020 Further input/output attempted before WAIT statement executed to ensure completion of previous READ.

1021 Attempt to access a record locked by another file in this process.

1022 Unable to extend indexed data set.

1023 Exclusive file closed while records still locked in a subtask

1024 Incorrect sequence of I/O operations on device-associated file.

1025 Insufficient virtual storage available to complete request.

1026 No position established in index data set.

1027 Record control interval already held in exclusive control.

1028 Requested record lies on an unmounted volume.

1029 Attempt to reposition in index data set failed.

1030 An error occurred during index upgrade on a index data set.

1031 Invalid sequential write attempted on index data set.

1040 A data set open for output used all available space.

1041 An attempt was made to write a record containing a record delimiter.

1042 Record in data set is not properly delimited.

1043 I/O error during CLOSE processing.

1062 Record length incorrect for RRDS file.

1068 VSAM server was not available.

1069 A deadlock was detected while attempting to lock a record.

1071 A retained lock reject has occurred while attempting to lock a record.

1094 Alternate index pointer invalid.

1102 An error occurred in storage management. Storage to be freed was pointed to by an invalid address.

1104 An internal error occurred in the library.

1105 Unable to create an object window.

1106 Insufficient space available to satisfy a storage allocation request.

1107 A problem occurred during free storage processing.

1301 F-factor in PICTURE specification was outside of the range of -128 to 127.

1302 PICTURE specification contained invalid character.

1303 F-factor contained invalid character.

1304 PICTURE specification contained invalid character.

1305 PICTURE specification contained invalid precision value.

1306 PICTURE specification contained too many overpunch characters.

1307 PICTURE specification contained precision value less than 1.

1308 Precision value in fixed decimal PICTURE specification exceeded limit.

1309 Precision value in float decimal PICTURE specification exceeded limit.

1310 PICTURE specification did not contain picture characters.

1311 Exponent in float PICTURE specification exceeded limit.

1312 Exponent in float PICTURE specification was missing.

1313 Exponent in PICTURE specification contained V character.

1314 Float PICTURE specification contained invalid character.

1315 PICTURE specification exceeded limit.

1316 PICTURE specification contained invalid delimiter.

Condition codes 1500 through 2000

1500 Computational error; short floating-point argument of SQRT built-in function is less than zero.

1501 Computational error; long floating-point argument of SQRT built-in function is less than zero.

1502 Computational error; extended floating-point argument of SQRT built-in function is less than zero.

1503 Computational error in LOG, LOG2, or LOG10 built-in function; extended floating-point argument is less than zero.

1504 Computational error in LOG, LOG2, or LOG10 built-in function; short floating-point argument is less than zero.

1505 Computational error in LOG, LOG2 or LOG10 built-in function; long floating-point argument is less than zero.

1506 Computational error in SIN, COS, SIND, or COSD built-in function; absolute value of short floating-point argument is too large. The limit depends on the representation as follows:

Representation	Limit
hexadecimal	pi*(2**18)
z ieee decimal	1s6

1507 Computational error in SIN, COS, SIND, or COSD built-in function; absolute value of long floating-point argument is too large The limit depends on the representation as follows:

Representation	Limit
hexadecimal	pi*(2**50)
z ieee binary	3.53711d15
z ieee decimal	1d15
i ieee binary	2**63

1508 Computational error; absolute value of short floating-point argument of TAN or TAND built-in function is too large The limit depends on the representation as follows:

Representation	Limit
hexadecimal	pi*(2**18)
z ieee decimal	1s6

1509 Computational error; absolute value of long floating-point argument of TAN or TAND built-in function is too large. The limit depends on the representation as follows:

Representation	Limit
hexadecimal	pi*(2**50)
z ieee binary	3.53711d15

Representation	Limit
z ieee decimal	1d15
i ieee binary	2**63

1514 Computational error; absolute value of short floating-point argument of ATANH built-in function >1.

1515 Computational error; absolute value of long floating-point argument of ATANH built-in function >1.

1516 Computational error; absolute value of extended floating-point argument of ATANH built-in function >1.

1517 Computational error in SIN, COS, SIND, or COSD built-in function; argument of extended floating-point argument is too large. The limit depends on the representation as follows:

Representation	Limit
hexadecimal	pi*(2**100)
z ieee binary	4.07802q33
z ieee decimal	1q33
i ieee binary	2**64

1518 Computational error; absolute value of short floating-point argument of ASIN or ACOS built-in function exceeds 1.

1519 Computational error; absolute value of long floating-point argument of ASIN or ACOS built-in function exceeds 1.

1520 Computational error; absolute value of extended floating-point argument of ASIN, ACOS built-in function exceeds 1.

1522 Computational error; absolute value of extended floating-point argument of TAN or TAND built-in function is too large The limit depends on the representation as follows:

Representation	Limit
hexadecimal	pi*(2**100)
z ieee binary	4.07802q33
z ieee decimal	1q33
i ieee binary	2**64

1523 Computational error; absolute value of real short floating-point argument of SINH or COSH built-in function is too large The limit depends on the representation as follows:

Representation	Limit
hexadecimal	175.366
z ieee decimal	2.233507s02

1524 Absolute value of real long floating-point argument of SINH or COSH argument is too large The limit depends on the representation as follows:

Representation	Limit
hexadecimal	175.366
z ieee binary	709.7827
z ieee decimal	8.864952608027075d02
i ieee binary	710.47

1525 Absolute value of real extended floating-point argument of SINH or COSH is too large, The limit depends on the representation as follows:

Representation	Limit
hexadecimal	175.366
z ieee binary	11354
z ieee decimal	1.41493853964484107282905574890354q4
i ieee binary	11357.56

1529 Computational error in SIN, COS, SIND, or COSD built-in function; absolute value of the real part of complex short floating-point argument greater is too large The limit depends on the representation as follows:

Representation	Limit
hexadecimal	pi*(2**18)
z ieee decimal	1s6

1530 Computational error in SIN, COS, SIND, or COSD built-in function; absolute value of the real part of complex long floating-point argument is too large. The limit depends on the representation as follows:

Representation	Limit
hexadecimal	pi*(2**50)
z ieee binary	3.53711d15
z ieee decimal	1d15
i ieee binary	2**63

1531 Computational error in SIN, COS, SIND, or COSD built-in function; absolute value of the real part of complex extended floating-point is too large. The limit depends on the representation as follows:

Representation	Limit
hexadecimal	pi*(2**100)
z ieee binary	4.07802q33
z ieee decimal	1q33
i ieee binary	2**64

1550 Computational error; during exponentiation, real short floating-point base is zero and integer exponent is not positive.

1551 Computational error; during exponentiation, real long floating-point base is zero and integer exponent is not positive.

1552 Computational error; during exponentiation, real short floating-point base is zero and the floating-point or noninteger exponent is not positive.

1553 Computational error; during exponentiation, real long floating-point base is zero and the floating-point or noninteger exponent is not positive.

1554 Computational error; during exponentiation, complex short floating-point base is zero and integer exponent is not positive.

1555 Computational error; during exponentiation, complex long floating-point base is zero and integer exponent is not positive.

1556 Computational error; during exponentiation, complex short floating-point base is zero and floating-point or noninteger exponent is not positive and real.

1557 Computational error; during exponentiation, complex long floating-point base is zero and floating-point or noninteger exponent is not positive and real.

1558 Computational error; complex short floating-point argument of ATAN or ATAND built-in function has value, respectively, of ±1I or ±1.

1559 Computational error; complex long floating-point argument of ATAN or ATAND built-in function has value, respectively, of ±1I or ±1.

1560 Computational error; during exponentiation, real extended floating-point base is zero and integer exponent not positive.

1561 Computational error; during exponentiation, real extended floating-point base is zero and floating-point or noninteger exponent is not positive.

1562 Computational error; during exponentiation, complex extended floating-point base is zero and integer exponent is not positive.

1563 Computational error; complex extended floating-point base is zero and floating-point or nonintegral exponent is not positive.

1564 Computational error; complex extended floating-point argument of ATAN or ATAND built-in function has value, respectively, of ±1I or ±1.

1568 Computational error EXP built-in function; absolute value of the imaginary part of the complex short floating-point argument is too large. The limit depends on the representation as follows:

Representation	Limit
hexadecimal	pi*(2**18)
z ieee decimal	1s6

1569 Computational error EXP built-in function; absolute value of the imaginary part of the complex long floating-point argument is too large. The limit depends on the representation as follows:

Representation	Limit
hexadecimal	pi*(2**50)
z ieee binary	3.53711d15
z ieee decimal	1d15
i ieee binary	2**63

1570 Computational error EXP built-in function; absolute value of the imaginary part of the complex extended floating-point argument is too large. The limit depends on the representation as follows:

Representation	Limit
hexadecimal	pi*(2**100)
z ieee binary	4.07802q33
z ieee decimal	1qd33
i ieee binary	2**64

1571 Computational error GAMMA or LOGGAMMA built-in function; real short floating point argument is too large. The limit for GAMMA depends on the representation as follows:

Representation	Limit
hexadecimal	57.5744
z ieee decimal	6.932968s01

The limit for LOGGAMMA depends on the representation as follows:

Representation	Limit
hexadecimal	4.2937*(10**73)
z ieee decimal	4.608910s94

1572 Computational error GAMMA or LOGGAMMA built-in function; real long floating point argument is too large. The limit for GAMMA depends on the representation as follows:

Representation	Limit
hexadecimal	57.5744
z ieee binary	171.624
z ieee decimal	2.053796629328708d02
i ieee binary	171.6243

The limit for LOGGAMMA depends on the representation as follows:

Representation	Limit
hexadecimal	4.2937*(10**73)
z ieee binary	2.559d305
z ieee decimal	1.138023083333461d382
i ieee binary	2.0d0**1014

1573 Computational error GAMMA or LOGGAMMA built-in function; real extended floating point argument is too large. The limit for GAMMA depends on the representation as follows:

Representation	Limit
hexadecimal	57.5744
z ieee binary	1755
z ieee decimal	2.12454995666246323632807135355444q3
i ieee binary	171.6243

The limit for LOGGAMMA depends on the representation as follows:

Representation	Limit
hexadecimal	4.2937*(10**73)
z ieee binary	1q4928
z ieee decimal	7.0727216522809330616880996925296 3q6140
i ieee binary	2.0q0**1014

1574 Computational error TANH built-in function; absolute value of the imaginary part of the complex short floating-point argument is too large. The limit depends on the representation as follows:

Representation	Limit
hexadecimal	pi*(2**18)
z ieee decimal	1s6

1575 Computational error TANH built-in function; absolute value of the imaginary part of the complex long floating-point argument is too large. The limit depends on the representation as follows:

Representation	Limit
hexadecimal	pi*(2**50)

Representation	Limit
z ieee binary	3.53711d15
z ieee decimal	1d15
i ieee binary	2**63

1576 Computational error TANH built-in function; absolute value of the imaginary part of the complex extended floating-point argument is too large. The limit depends on the representation as follows:

Representation	Limit
hexadecimal	pi*(2**100)
z ieee binary	4.07802q33
z ieee decimal	1q33
i ieee binary	2**64

1577 Computational error in LOG, LOG2, or LOG10 built-in function; real short floating-point argument equal to zero.

1578 Computational error in LOG, LOG2, or LOG10 built-in function; real long floating-point argument equal to zero.

1579 Computational error in LOG, LOG2, or LOG10 built-in function; real extended floating-point argument equal to zero.

1611 Computational error; real short floating-point argument for EXP built-in function is too large. The limit depends on the representation as follows:

Representation	Limit
hexadecimal	174.673
z ieee decimal	2.233507s02

1612 Computational error; real long floating-point argument for EXP built-in function is too large. The limit depends on the representation as follows:

Representation	Limit
hexadecimal	174.673
z ieee binary	709.7827
z ieee decimal	8.864952608027075d02
i ieee binary	710.47

1613 Computational error; real extended floating-point argument for EXP built-in function is too large. The limit depends on the representation as follows:

Representation	Limit
hexadecimal	174.673
z ieee binary	11354
z ieee decimal	1.41493853964484107282905574890354q4
i ieee binary	11357.56

1729 Computational error; during exponentiation, real short floating-point base is zero and real short floating-point exponent is not positive or zero.

1730 Computational error; during exponentiation, real long floating-point base is zero and real long floating-point exponent is not positive or zero.

1754 Computational error; during exponentiation for a complex short floating-point base with a complex short floating-point exponent, an argument exceeded the limit.

1755 Computational error; during exponentiation for a complex long floating-point base with a complex long floating-point exponent, an argument exceeded the limit.

1756 Computational error; during exponentiation for a complex extended floating-point base with a complex extended floating-point exponent, an argument exceeded the limit.

1853 Computational error in TAN or TAND; for complex short floating-point argument, absolute value of the real part of argument is too large. The limit depends on the representation as follows:

Representation	Limit
hexadecimal	pi*(2**18)
z ieee decimal	1s6

1854 Computational error in TAN or TAND; for complex long floating-point argument, absolute value of the real part of argument is too large. The limit depends on the representation as follows:

Representation	Limit
hexadecimal	pi*(2**50)
z ieee binary	3.53711d15
z ieee decimal	1d15
i ieee binary	2**63

1855 Computational error in TAN or TAND; for complex extended floating-point argument, absolute value of the real part of argument is too large. The limit depends on the representation as follows:

Representation	Limit
hexadecimal	pi*(2**100)
z ieee binary	4.07802q33
z ieee decimal	1q33
i ieee binary	2**64

1914 Computational error; absolute value of imaginary part of complex short floating-point argument of SINH or COSH built-in function is too large. The limit depends on the representation as follows:

Representation	Limit
hexadecimal	pi*(2**18)
z ieee decimal	1s6

1915 Computational error; absolute value of the imaginary part of complex long floating-point argument of SINH or COSH built-in is too large. The limit depends on the representation as follows:

Representation	Limit
hexadecimal	pi*(2**50)
z ieee binary	3.53711d15
z ieee decimal	1d15
i ieee binary	2**63

1916 Computational error; absolute value of the imaginary part of complex extended floating-point argument of SINH or COSH built-in is too large. The limit depends on the representation as follows:

Representation	Limit
hexadecimal	pi*(2**100)
z ieee binary	4.07802q33
z ieee decimal	1q33
i ieee binary	2**64

1960 Computational error in SQRT; real short floating-point argument is equal to zero.

1961 Computational error in SQRT; real long floating-point argument is equal to zero.

1962 Computational error in SQRT; real extended floating-point argument is equal to zero.

Condition codes 2001 through 2500

2002 WAIT statement cannot be executed because of restricted system facility.

2050 WAIT statement that causes permanent wait encountered.

2101 Greenwich mean time was not available for the RANDOM built-in function.

2102 An invalid seed value was detected in the RANDOM built-in function. The random number was set to -1.

2103 Local time was unavailable.

2104 The value of *y* in the SECSTODATE, DAYS, DAYSTODATE, or DATETIME built-in function contained an invalid picture string specification.

2105 The value of *x* in the DAYS built-in function contained an invalid day value; the valid range is 15 October 1582 to 31 December 9999.

2106 The value of *x* in the DAYS built-in function contained an invalid month value; the valid range is October 1582 to December 9999.

2107 The value of *x* in the DAYS built-in function contained an invalid year value; the valid range is 1582 to 9999.

2108 The value of *x* in the DAYSTODATE built-in function was outside the supported range; the valid range is from 1 to 3,074,324.

2109 The value of *x* in the SECSTODATE built-in function was outside the supported range; the valid range is from 86,400 to 265,621,679,999.999.

2110 The value of *x* in the DAYSTODATE built-in function could not be converted to a valid Japanese or Republic of China Era.

2111 The difference between the current local time and the Greenwich Mean Time was unavailable.

2112 The value of *x* in the SECS or DAYS built-in function was outside the supported range; the valid range is from 15 October 1582 to 31 December 9999.

2113 The value of *x* in the SECS built-in function contained an invalid seconds value; the valid range is from 0 to 59.

2114 The value of *x* in the SECS built-in function contained an invalid minutes value; the valid range is from 0 to 59.

2115 The value of *x* in the SECS built-in function contained an invalid hour value; the valid range is from 0 to 23 or from 0 to 12 (if the AP field is present).

2116 The value of *x* in the DAYS built-in function did not match the given picture specification.

2117 The value of *x* in the SECS built-in function did not match the given picture specification.

2118 The date string returned by the DAYSTODATE built-in function was truncated.

2119 The timestamp returned by the DATETIME or SECSTODATE built-in function was truncated.

2120 The value of *x* in the SECSTODATE or DATETIME built-in function contained an invalid value for the number of seconds with the range of supported Japanese or Republic of China Eras.

2121 Insufficient data was passed to the DAYS or SECS built-in function; the picture string did not contain enough information.

2122 The value of *x* in the SECS or DAYS built-in function contained an invalid Era name.

2165 Computational error GAMMA or LOGGAMMA built-in function; real short floating point argument is less than or equal to zero.

2166 Computational error GAMMA or LOGGAMMA built-in function; real long floating point argument is less than or equal to zero.

2167 Computational error GAMMA or LOGGAMMA built-in function; real extended floating point argument is less than or equal to zero.

2171 Real short floating-point argument greater than limit.

2172 Real long floating-point argument greater than limit.

2173 Real extended floating-point argument greater than limit.

2403 Computational error; real extended floating point argument of GAMMA or LOGGAMMA built-in function was less than or equal to zero.

2404 Computational error; real extended floating point argument of GAMMA or LOGGAMMA built-in function was equal to zero.

2413 Computational error; complex short floating-point argument in LOG, LOG2, or LOG10 built-in function was zero.

2414 Computational error; complex long floating-point argument in LOG, LOG2, or LOG10 built-in function was zero.

2415 Computational error; complex extended floating-point argument in LOG, LOG2, or LOG10 built-in function was zero.

2504 Real short floating-point argument greater than allowed value for data type.

2505 Real long floating-point argument greater than allowed value for data type.

2506 Real extended floating-point argument greater than allowed value for data type.

Condition codes 3000 through 4000

3000 Field width, number of fractional digits, and number of significant digits (w, d, and s) specified for E-format item in edit-directed input/output statement do not allow transmission without loss of significant digits or sign.

3002 MEMCONVERT built-in returned a bad return code.

3003 No room for shift-in after Unicode conversion.

3006 Picture description of target does not match non-character-string source.

3009 A mixed-character string contained a shift-out, then ended before a shift-in was found.

3010 During processing of a mixed-character constant, one of the following occurred:

- A shift-in present in the SBCS portion.
- A shift-out present in the graphic (double-byte) portion. (A shift-out cannot appear in either byte of a graphic character).
- A shift-in present in the second byte of a graphic character.

3011 MPSTR built-in function contains an invalid character (or a null function string, or only blanks) in the expression that specifies processing rules. (Only V, v, S, s, and blank are valid characters.)

3013 An assignment attempted to a graphic target with a length greater than 16,383 characters (32,766 bytes).

3014 A graphic or mixed string did not conform to the continuation rules.

3015 A *X* or *GX* constant has an invalid number of digits.

3016 Improper use of graphic data in stream I/O. Graphic data can only be used as part of a variable name or string.

3018 Invalid UTF-8 data was detected.

3019 An invalid byte 2 in a UTF-8 character was detected.

3020 An invalid byte 3 in a UTF-8 character was detected.

3021 An invalid byte 4 in a UTF-8 character was detected.

3022 An incomplete UTF-8 character was detected.

3023 Invalid UTF-16 data was detected.

3024 An incomplete UTF-16 character was detected.

3025 USUBSTR reference is invalid.

3500 Error detected by the operating system while processing WAIT statement.

3501 Error detected by the operating system while processing DETACH statement.

3502 Error detected by the operating system while processing ATTACH statement.

3503 Error detected by the operating system while processing STOP statement.

3504 ATTACH statement being processed in POSIX(OFF) environment.

3797 Attempt to convert to or from graphic data.

3798 ONCHAR, ONSOURCE, or ONGSOURCE pseudovariable used out of context.

3799 The source was not modified in the CONVERSION ON-unit. Retry was not attempted. An ON-unit was entered as a result of the CONVERSION condition being raised by an invalid character in the string being converted. The character was not corrected in an ON-unit using the ONSOURCE, ONGSOURCE, or ONCHAR pseudovariables.

3800 Length of data aggregate exceeds system limit of 2**24 bytes.

3804 Array initialization exceeded maximum depth of iteration.

3808 Aggregate cannot be mapped in COBOL or FORTRAN.

3809 A data aggregate exceeded the maximum length.

3810 An array has an extent that exceeds the allowable maximum.

3901 Attempt to invoke process using a process variable that is already associated with an active process.

3904 Event variable referenced as argument to COMPLETION pseudovariable while already in use for a DISPLAY statement.

3906 Assignment to an event variable that is already active.

3907 Attempt to associate an event variable that is already associated with an active process.

3908 Query of installation default of maximum number of threads failed.

3909 Attempt to create a subtask (using CALL statement) when insufficient main storage available.

3910 Attempt to attach a process (using CALL statement) when number of active processes was already at limit defined by ISASIZE parameter of EXEC statement.

3911 WAIT statement in ON-unit references an event variable already being waited for in process from which ON-unit was entered.

3912 Attempt to execute CALL with TASK option in block invoked while executing PUT FILE(SYSPRINT) statement.

3913 CALL statement with TASK option specifies an unknown entry point.

3914 Attempt to call FORTRAN or COBOL routines in two processes simultaneously.

3915 Attempt to call a process when the multitasking library was not selected in the link-edit step.

3920 An out-of-storage abend occurred.

3951 Call to initialize wait failed.

3952 Call to perform wait failed.

3953 Call to cancel a subtask failed.

3954 Call to support PL/I EXCLUSIVE files failed.

Condition codes 4001 through 9999

4001 Attempt to assign data to an unallocate CONTROLLED variable occurred on a GET DATA statement.

4002 Attempt to output an unallocate CONTROLLED variable occurred on a PUT DATA statement.

4003 Attempt to assign from an unallocate CONTROLLED variable occurred on a PUT DATA statement with the STRING option.

5050 Too many digits specified in JSON floating-point number.

5051 Too many digits specified in JSON fixed-point number.

5052 Invalid value type in JSON text.

5053 Conversion from UTF-8 to character failed.

5054 Source in JSON assignment to BIT is invalid.

5055 Conversion from UTF-8 to UTF-16 failed.

5056 String in JSON text is too long.

5057 Characters after \u are not valid hexadecimal digits.

5058 Hexadecimal characters specify an invalid UTF surrogate pair.

5059 Invalid escape character in JSON text.

5060 Only valid value starting with t in JSON text is true.

5061 Only valid value starting with f in JSON text is false.

5062 Only valid value starting with n in JSON text is null.

5063 JSON text ends prematurely.

5064 Number does not conform to the rules of JSON syntax.

5065 Name in JSON source does not match that in the target.

5066 The JSON values true and false may be assigned only to NONVARYING BIT.

5067 JSON text contains invalid UTF-8 characters.

5068 Objects and arrays in the JSON text are nested too deeply.

5069 Next significant character in the JSON text should be an opening bracket, [.

5070 Next significant character in the JSON text should be a closing bracket,].

5071 Next significant character in the JSON text should be an opening brace, {.

5072 Next significant character in the JSON text should be a closing brace, }.

5073 Next significant character in the JSON text should be a comma (,).

5074 Next significant character in the JSON text should be a double quotation mark (").

5075 Next significant character in the JSON text should be a colon (:).

5076 Next significant character in the JSON text should be the start of a JSON value.

5077 Next significant character in the JSON text should be a closing bracket,], or the start of a JSON value.

5078 Next significant character in the JSON text should be a double quotation mark (") or a closing brace, }.

5079 Next significant character in the JSON text should be a comma (,) or a closing bracket,].

5080 Next significant character in the JSON text should be a comma (,) or a closing brace, }.

8091 Operation exception.

8092 Privileged operation exception.

8093 EXECUTE exception.

8094 Protection exception.

8095 Addressing exception.

8096 Specification exception.

8097 Data exception.

8098 Insufficient stack storage

9002 Attempt to execute GO TO statement referencing label in an inactive block.

9003 Attempt to execute a GO TO statement to a nonexistent label constant.

9004 RETURN without return value attempted from procedure with RETURNS attribute.

9005 RETURN with return value attempted from procedure without RETURNS attribute.

9050 Program terminated by an abend.

9051 An error occurred in CICS. It is highly likely that parameters, particularly pointers, specified on the EXEC CICS command do not point at storage owned by the PL/I program. The ERROR on-unit is not given control. When the TEST run-time option is in effect, PLITEST allows the user to examine variables, etc. but the execution cannot be continued.

9200 Program check in SORT/MERGE program.

9201 SORT not supported in CMS.

9202 RECORD TYPE string missing in the PLISRTx call.

9203 Incorrect record type specified in the PLISRTx call.

9204 LENGTH= missing from RECORD TYPE string specification in the PLISRTB or PLISRTD call.

9205 Length specified in the LENGTH= parameter of the PLISRTx call is not numeric.

9206 Incorrect return code received from E15 or E35 data-handling routine.

9207 DFSORT failed with the return code displayed in the message.

9208 PLISRTx invoked in an environment other than ADMVS.

9209 Fetch of SMARTSort failed.

9210 DD for SORT input data set invalid.

9211 DD for SORT output data set invalid.

9212 DD for SORT data set missing LRECL or LENGTH.

9213 DD for SORT data set must specify a TYPE.

9214 CALL PLISRTx statement missing a SORT FIELDS string.

9215 SORT FIELDS parameter of CALL PLISRTx statement specified too many fields.

9216 SORT FIELDS parameter of CALL PLISRTx statement contained invalid start, length fields, or both.

9217 SORT FIELDS parameter of CALL PLISRTx statement contained invalid form.

9218 SORT FIELDS parameter of CALL PLISRTx statement contained invalid sequence.

9249 Routine cannot be released.

9250 Procedure to be fetched cannot be found.

9251 Permanent transmission error when fetching a procedure.

9252 FETCH/RELEASE not supported in CMS.

9253 PLITEST unavailable.

9254 Attempt made to release load module containing non-PL/I high level language programs.

9255 SORT FIELDS parameter of CALL PLISRTx statement contained invalid sequence.

9258 Routine compiled with NORENT cannot fetch routine compiled with RENT.

9999 A failure occurred in invocation of a Language Environment service.

Notices

This information was developed for products and services offered in the U.S.A. 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 Corporation
J74/G4
555 Bailey Avenue
San Jose, CA 95141-1099
U.S.A.

For license inquiries regarding double-byte (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.
3-2-12, Roppongi, Minato-ku, Tokyo 106-8711

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 publication 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.

© Copyright IBM Corp. 1999, 2018

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.

Intel is a registered trademark of Intel Corporation in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

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

Pentium is a registered trademark of Intel Corporation in the United States and other countries.

Unicode is a trademark of the Unicode Consortium.

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

Other company, product or service names may be the trademarks or service marks of others.

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

Bibliography

PL/I publications

Enterprise PL/I for z/OS

Programming Guide, GI11-9145
Language Reference, SC14-7285
Messages and Codes, GC14-7286
Compiler and Run-Time Migration Guide, GC14-7284

PL/I for MVS & VM

Installation and Customization under MVS, SC26-3119
Language Reference, SC26-3114
Compile-Time Messages and Codes, SC26-3229
Diagnosis Guide, SC26-3149
Migration Guide, SC26-3118
Programming Guide, SC26-3113
Reference Summary, SX26-3821

PL/I for AIX

Programming Guide, SC14-7319
Language Reference, SC14-7320
Messages and Codes, GC14-7321
Installation Guide, GC14-7322

Related publications

Db2 and z/OS

Administration Guide, SC19-2968
Application Programming and SQL Guide, SC19-2969
Command Reference, SC19-2972
Messages, GC19-2979
Codes, GC19-2971
SQL Reference, SC19-2983
See also the Information Center: publib.boulder.ibm.com/infocenter/dzichelp/v2r2/topic/com.ibm.db2z10.doc/src/alltoc/db2z_10_prodhome.htm
LOBs with Db2 for z/OS: Stronger and Faster, SG24-7270

DFSORT™

Application Programming Guide, SC33-4035
Installation and Customization, SC33-4034

IMS/ESA®

Application Programming: Database Manager, SC26-8015
Application Programming: Database Manager Summary, SC26-8037
Application Programming: Design Guide, SC26-8016

© Copyright IBM Corp. 1999, 2018

Application Programming: Transaction Manager, SC26-8017

Application Programming: Transaction Manager Summary, SC26-8038

Application Programming: EXEC DL/I Commands for CICS and IMS™, SC26-8018

Application Programming: EXEC DL/I Commands for CICS and IMS Summary, SC26-8036

TXSeries for Multiplatforms

Encina Administration Guide Volume 2: Server Administration, SC09-4474

Encina SFS Programming Guide, SC09-4483

See also the Information Center: publib.boulder.ibm.com/infocenter/txformp/v7r1/index.jsp

z/Architecture

Principles of Operation, SA22-7832

z/OS Language Environment

Concepts Guide, SA22-7567

Debugging Guide, GA22-7560

Run-Time Messages, SA22-7566

Customization, SA22-7564

Programming Guide, SA22-7561

Programming Reference, SA22-7562

Run-Time Application Migration Guide, GA22-7565

Writing Interlanguage Communication Applications, SA22-7563

z/OS MVS

JCL Reference, SA22-7597

JCL User's Guide, SA22-7598

System Commands, SA22-7627

z/OS UNIX System Services

z/OS UNIX System Services Command Reference, SA22-7802

z/OS UNIX System Services Programming: Assembler Callable Services Reference, SA22-7803

z/OS UNIX System Services User's Guide, SA22-7801

Unicode® and character representation

OS/390 Support for Unicode: Using Conversion Services, SC33-7050

Readers' Comments — We'd Like to Hear from You

Enterprise PL/I for z/OS
IBM Developer for z Systems PL/I for Windows
Messages and Codes
Version 5 Release 2

Publication No. GC27-8950-01

We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state on this form.

Comments:

Thank you for your support.

Send your comments to the address on the reverse side of this form.

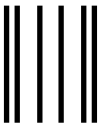
If you would like a response from IBM, please fill in the following information:

Name	Address
Company or Organization	
Phone No.	Email address

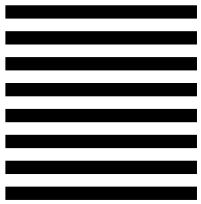
Readers' Comments — We'd Like to Hear from You
GC27-8950-01

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape



NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES



BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM
H150/090
555 Bailey Avenue
San Jose, CA
USA 95141-1099

Fold and Tape | **Please do not staple** | Fold and Tape

GC27-8950-01

Cut or Fold
Along Line

Product Number: 5655-PL5

Printed in USA

GC27-8950-01